

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática



**DESENVOLVIMENTO DE UMA FRAMEWORK DE
BUSINESS PERFORMANCE MANAGEMENT**

João Tiago Ribeiro Mendes Natálio

PROJECTO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

2011

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática



**DESENVOLVIMENTO DE UMA FRAMEWORK DE
BUSINESS PERFORMANCE MANAGEMENT**

João Tiago Ribeiro Mendes Natálio

PROJECTO

Trabalho orientado pelo Prof. Doutor Pedro Alexandre de Mourão Antunes
e co-orientado pelo Engenheiro Nuno Miguel Pombo Rodrigues

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Engenharia de Software

2011

Resumo

Este documento relata detalhadamente todo o trabalho executado no estágio desenvolvido durante nove meses na empresa GEDI (Gabinete de Estudos e Divulgação Informática, SA) pelo aluno João Tiago Ribeiro Mendes Natálio, no âmbito do Mestrado em Engenharia Informática da Faculdade de Ciências da Universidade de Lisboa. O trabalho consta num projecto de desenvolvimento de software onde se irá desenvolver uma *framework* de *business intelligence* focada na área de performance de negócio, uma área importante no apoio á decisão nas organizações. A *framework* de BPM (*Business Performance Management*) visa ajudar as organizações no uso eficiente dos seus recursos financeiros, capital humano, bem como outros recursos.

Procurou-se desenvolver um sistema flexível, capaz de reunir informação de várias fontes de dados através da aplicação de técnicas de ETL (extração, transformação e carregamento de dados) para a construção de *Data Warehouses* e que funciona de forma integrada no Sistema Integrado de Apoio á Gestão para a Administração Pública desenvolvido pela GEDI. As funcionalidades de análise e exploração de dados já existentes no SIAG-AP foram actualizadas para suportar a exploração de fontes de dados seleccionadas pelos utilizadores, funcionalidades essas que actuam sobre o *Mondrian*, um motor de OLAP (*Online Analytical Processing*) desenvolvido pela empresa *Pentaho*.

O projecto desenvolvido fornece várias funcionalidades de exploração e análise de dados, assim como coloca à disposição métricas e indicadores de gestão relativos a processos de negócio.

Palavras-chave: j2ee, hibernate, spring, data warehouse, etl, bi, bpm, olap, mondrian

Abstract

This document describes in detail all work performed on stage during nine months in the company GEDI (Gabinete de Estudos e Divulgação Informática, SA) by student João Tiago Ribeiro Mendes Natálio in the Computer Engineering Project at Faculdade de Ciências da Universidade de Lisboa. This work involves a software development project which will create a *Business Intelligence Framework* focused on the area of *Business Performance Management* and which will serve as an important tool for decision support to the organizations. The *Business Performance Management Framework* aims to help organizations in the efficient use of their financial resources, human capital, materials and other resources.

We developed a flexible system capable of gathering information from several different data sources through the application of ETL (*Extract, Transform and Load*) techniques for the construction of several *Data Warehouses*, which works seamlessly with the Sistema Integrado de Apoio á Gestão para a Administração Publica developed by GEDI. The data analysis and exploration features of SIAG-AP were upgraded to support data exploration over data sources selected by the end users, features that rely on *Mondrian*, an OLAP (*Online Analytical Processing*) engine developed by *Pentaho*.

The developed project provides various features of exploration and data analysis, as well as providing metrics and management indicators related to business processes.

Keywords: j2ee, hibernate, spring, data warehouse, etl, bi, bpm, olap, mondrian

Conteúdo

Lista de Figuras	xi
Lista de Tabelas	xiii
1 Introdução	1
1.1 Motivação	1
1.2 Objectivos	2
1.3 Estrutura do documento	3
2 A Organização	5
2.1 Estrutura da Organização	6
2.2 Integração na Equipa de Desenvolvimento	6
3 Planeamento	9
3.1 Tarefas no Projecto	9
3.1.1 Planeamento e Acompanhamento do Projecto	9
3.1.2 Levantamento de Requisitos	9
3.1.3 Análise de Requisitos	9
3.1.4 Criação do Módulo SiagBI Independente	10
3.1.5 Independência de SiagBI das Fontes de Dados	10
3.1.6 Implementação da Componente de ETL	10
3.1.7 Testes	10
3.1.8 Documentação para Utilizador Final	10
3.1.9 Criação de Relatório Preliminar	10
3.1.10 Criação de Relatório Final	11
3.2 Planeamento das Tarefas	11
3.2.1 Cronograma Inicial do Projecto	11
3.2.2 Cronograma Final do Projecto	11
3.3 Metodologia de Desenvolvimento	12
4 Análise e Pesquisa	15
4.1 Business Intelligence	15

4.2	Business Performance Management	16
4.3	Data Warehouse	16
4.3.1	Arquitectura	17
4.3.2	Modelo de Dados	17
4.3.3	Princípios de Design	19
4.4	Extract, Transform and Load	21
4.5	Online Analytical Processing	22
4.6	Levantamento de Requisitos	23
4.6.1	Requisitos Funcionais	24
4.6.2	Requisitos Não-Funcionais	24
5	Estudo de Mercado	27
5.1	Pentaho Business Intelligence Suite	27
5.1.1	Pentaho Business Intelligence Server	30
5.2	Ferramentas de Extract, Transform and Load	33
5.2.1	Pentaho Data Integration	33
5.2.2	Talend Open Studio	37
5.2.3	CloverETL	39
5.3	Conclusão da Análise	40
6	Arquitectura e Tecnologias Utilizadas	43
6.1	O Sistema Integrado de Apoio à Gestão	43
6.1.1	A antiga arquitectura	43
6.1.2	A nova arquitectura	44
6.2	Módulo SiagBI	46
6.2.1	Arquitectura	46
6.2.2	Modelo de Dados	48
6.3	Módulo SiagETL	50
6.3.1	Arquitectura	50
6.3.2	Modelo de Dados	51
6.4	Tecnologias Utilizadas	52
6.4.1	<i>Java Messaging System</i>	52
6.5	Testes	53
6.5.1	Testes sobre o módulo SiagBI	54
6.5.2	Testes sobre o módulo SiagETL	54
7	Trabalho Realizado	57
7.1	Criação do módulo SiagBI	57
7.1.1	Organização dos <i>packages</i> Java	57
7.1.2	Passagem das funcionalidades de BI para o novo módulo	58

7.1.3	Independência de Fontes de Dados	59
7.1.4	Actualização das Bibliotecas <i>Mondrian</i> e <i>JPivot</i>	60
7.1.5	Criação da interface GWT	61
7.2	Criação do módulo SiagETL	62
7.2.1	Organização dos <i>packages</i> Java	63
7.2.2	Integração do <i>Pentaho Data Integration</i>	63
7.2.3	Calendarização de Execução de Transformações ETL	65
7.2.4	Criação das interfaces GWT	67
8	Conclusão	71
8.1	Trabalho Futuro	71
8.2	Conclusão	72
	Bibliografia	82
	Anexos	82

Lista de Figuras

2.1	Âmbito das funcionalidades do sistema SIAG-AP	5
3.1	Cronograma do planeamento inicial do projecto	11
3.2	Cronograma do execução do projecto	11
4.1	Camadas da arquitectura de um Data Warehouse	17
4.2	Exemplo de um esquema em estrela	18
5.1	Modelo alto-nível dos componentes que constituem a <i>Pentaho Business Intelligence Suite</i>	28
5.2	Modelo alto-nível dos componentes que constituem o <i>Pentaho BI Server</i>	31
5.3	Esquema do funcionamento do Pentaho Data Integration	34
5.4	Modelo conceptual dos artefactos produzidos pelo Pentaho Data Integration	35
5.5	Modelo da integração das diferentes ferramentas do Kettle	36
5.6	Modelo resultante do exemplo ETL pelo Talend Open Studio	38
5.7	Modelo resultante do exemplo ETL pelo CloverETL	40
6.1	Diagrama do padrão Model-View-Controller	44
6.2	Diagrama UML de uma classe que implementa o padrão Singleton	46
6.3	Diagrama UML exemplar do módulo de criação de modelos de base de dados do SiagBI	47
6.4	Modelo de Dados do módulo SiagBI	49
6.5	Modelo de Dados do módulo SiagETL	52
7.1	Novo ecrã de criação de fontes de dados	61
7.2	Novo ecrã de criação de cubos	62
7.3	Modelo BPM do processo <i>ETLScheduler</i>	66
7.4	Interface de gestão de Transformações ETL no sistema SIAG-AP	67
7.5	<i>Dashboard</i> de acesso ao servidor <i>Carte</i>	68

Lista de Tabelas

4.1	Tabela de requisitos não-funcionais do projecto.	25
-----	--	----

Capítulo 1

Introdução

Vivemos numa realidade imperada pela inteligência e suportada por constantes fluxos de informação, fluxos esses que nunca antes foram tão volumosos, complexos e essenciais à nossa vida actual. E para acompanhar esta grande velocidade de evolução dos sistemas de informação, existe uma necessidade cada vez maior de ferramentas que nos permitam obter controlo sobre esse conhecimento difuso e mutável.

Uma vez que o mundo do negócio não é excepção a esta evolução, é neste panorama que surge software de suporte à *Business Intelligence*[60], desenvolvido para analisar e transformar enormes quantidades de dados para que se possa mais facilmente tomar proveito de todo o conhecimento existente dentro de uma organização. Neste capítulo iremos introduzir o nosso projecto, assim como iremos abordar o seu contexto no mundo do negócio e os seus objectivos a longo prazo.

1.1 Motivação

Na elaboração de um projecto com estas características existem vários requisitos a que um programador tem de dar a devida atenção. Desde logo, a adequação das necessidades do cliente à agilidade de manipulação da informação. Como sabemos, a quantidade de dados a que uma organização tem acesso é apenas útil se a ela se conseguir aceder de forma ágil e dentro de um determinado âmbito, alinhado com os desejos de quem analisa a informação. Outro elemento fundamental é conceber o filtramento da informação relevante no interior de um grande conjunto de dados e integrá-lo num processo de análise versátil, personalizado e participativo. No entanto, este processo torna-se obsoleto como mecanismo de apoio à decisão das organizações a partir do momento em que não consegue acompanhar a velocidade de evolução do negócio. O tempo que se demora a analisar e consumir a informação é determinante na capacidade de endereçar prontamente problemas de indústria reais e actuais.

Outro aspecto importante a ter em conta é que a informação muitas vezes se encontra dispersa, tornando a actividade de análise difícil, uma vez que envolve várias fontes

de informação sujeitas a um tratamento exaustivo, dispendioso e propício a erros por parte dos analistas. Este tratamento é tipicamente feito em ferramentas desactualizadas, muitas vezes operadas dispersa e manualmente, face a este novo panorama de gestão de informação e performance de negócio. Os consumidores de informação dentro de uma organização são tipicamente gestores de negócios com conhecimento mais focados nas áreas de gestão e finanças. Este perfil, conjugado com a referida dispersão e heterogeneidade da informação, evidenciam como mais um requisito deste tipo de produtos, a necessidade de uma abstracção da implementação física dos sistemas de informação. Esta abstracção vai de certa forma permitir aos seus utilizadores tirarem proveito da informação sem estar dependentes das equipas de TI para extrair e manipular a informação de que necessitam.

De acordo com o IBM Global CIO Study[12] existe uma relação directa entre o sucesso das organizações e o recurso a processos de análise de negócio informatizados como mecanismo de apoio à decisão. São estes processos que permitem otimizar o negócio no seu uso racional de recursos e aumentar o poder de investimento das empresas. É neste âmbito que a GEDI (Gabinete de Estudos e Divulgação Informática, SA[8]) procura expandir o seu produto principal, o sistema SIAG-AP (Sistema Integrado de Apoio à Gestão para a Administração Pública[34]) de forma a integrar um novo módulo de BPM (*Business Performance Management*[61]).

1.2 Objectivos

Este projecto tem como principal objectivo a criação de uma *framework* de BPM que irá funcionar em complementaridade à *framework* já existente do SIAG.

Este passo visa criar uma nova metodologia de desenvolvimento que irá ser adoptada em todos os futuros módulos do SIAG-AP assim como progressivamente nos módulos já integrados no sistema. Pretende-se então estabelecer uma metodologia orientada aos processos de negócio (*Business Driven Development*[59]) que passa por uma separação lógica dos diferentes módulos que compõe a *framework*, procurando atingir vários objectivos a longo prazo: promover a reutilização de código; garantir a agilidade e flexibilidade na implementação de novos módulos; estabelecer a independência de funcionamento dos diferentes módulos e, consequentemente, a preparação para resolver problemas futuros relacionados com a limitação da escalabilidade do sistema e permitir ainda adaptar as funcionalidades do SIAG-AP de acordo com as necessidades dos clientes para os quais é distribuído.

A *framework* de BPM irá atender pedidos vindos da aplicação SIAG-AP e também necessita de ter a possibilidade de atender pedidos de sistemas ERP (*Enterprise Resource Planning*[75]) de outras organizações, mantendo como boa prática a interoperabilidade. Iremos extrair algumas funcionalidades de BI já implementadas dentro do SIAG-AP e re-

ver os processos de BPM já implementados de acordo com os novos requisitos definidos.

Esta *framework* deverá ainda integrar novas mecânicas de ETL (*Extract, Transform and Load*[76]) que permitam aos utilizadores recolher dados de diversas fontes de informação e que permitam uma construção assistida de *Data Warehouses*[72]). Deverá também implementar métodos e padrões para a análise, limpeza e compilação de dados de maneira a simplificar a apresentação da informação a agentes de negócio. Entre estas técnicas, recorreremos à utilização de portais web com *dashboards*[66] personalizados pelos utilizadores, podendo estes ser compostos por relatórios, gráficos e diversos indicadores.

1.3 Estrutura do documento

Este documento está organizado da seguinte forma:

- Capítulo 2 — Informação sobre a GEDI, a sua estrutura como organização e a minha integração nos quadros empresa.
- Capítulo 3 — Planeamento do projecto, descrição das tarefas a ser executadas e calendarização do projecto.
- Capítulo 4 — Análise do problema, pesquisa das tecnologias e soluções encontradas para a realização do projecto.
- Capítulo 5 — Aspectos importantes do estudo de mercado efectuado sobre a plataforma de BPM da *Pentaho*[36] e sobre algumas ferramentas de ETL[76] para integração no projecto.
- Capítulo 6 — Arquitectura e tecnologias envolvidas na *framework* SIAG e integração dos módulos *SiagBI* e *SiagETL*.
- Capítulo 7 — Trabalho realizado ao longo dos nove meses de estágio, incluindo a extracção das funcionalidades BI já existentes no sistema SIAG-AP para o novo servidor de BI e a criação do novo servidor de ETL, assim como a construção das respectivas interfaces.
- Capítulo 8 — Conclusões finais do projecto e discussão de dificuldades encontradas durante a sua execução, assim como de trabalho futuro.

Capítulo 2

A Organização

A GEDI[8] é uma pequena empresa com 30 colaboradores que se encontra no mercado desde 1984, dedicada à concepção, desenvolvimento e comercialização de soluções informáticas centradas na gestão das organizações no sector público. Estas soluções foram concebidas à medida da evolução das necessidades de administração dos organismos públicos, sendo progressivamente encapsuladas num único produto, o SIAG-AP[34]. Este sistema procura assim endereçar numa abordagem integrada e global a problemática dos diversos sectores de gestão, conforme especificados na figura seguinte:

Vertente Estratégica - Funcionalidades para a gestão de topo					
Planeamento	Visão	Carta de Missão	Plano Plurianual de Actividades	Orçamento Plurianual	
Balanced Scorecard [BSC]	Estratégica	Mapa Estratégico	Objectivos Estratégicos	Iniciativas Estratégicas	
Avaliação do Desempenho	SIADAP 123 QUAR	Avaliação Organizacional por objectivos		Avaliação das Pessoas por objectivos	
Suporte à gestão - Funcionalidades para a gestão operacional					
Logística	Compras	Vendas		Stocks	
Gestão Patrimonial	Inventariação	Administração		Abates	
Gestão Financeira	POCP/ Planos Sectoriais	Gestão de Terceiros	Centros Analíticos	Tesouraria	Prestação de Contas
Pessoas	Gestão de Carreiras	Mapas de Pessoal	Processamento de Remunerações e Outros Abonos	Gestão do Capital Humano	Balanço Social
Informação	Registo de Informação	Classificação	Encaminhamento e Tratamento	Arquivo Electrónico	
Optimização do negócio - Funcionalidades para a melhoria da performance					
Business Intelligence	Conhecimento Organizacional Análise por Diversos Segmentos		Dashboards	Alertas	
BPM Business Process Management	Orquestração de Processos		Desenvolvimento de Novos Processos	Monitorização	
RAD Rapid Application Development	Personalização do Sistema	Desenvolvimento de Novas Funcionalidades, Designadamente de negócio		Integração com Outros Sistemas utilizando WebServices	
Portal do Utilizador	Desenho de Portais Específicos		Integração com Outros Portais		

Figura 2.1: Âmbito das funcionalidades do sistema SIAG-AP

Os projectos da GEDI procuram configurar o seu produto na adaptação constante às novas reformas públicas que vão surgindo, procurando não só alcançar os requisitos impostos pela respectiva legislação como também alcançar um nível elevado de maturidade das empresas.

2.1 Estrutura da Organização

Trata-se de uma organização matricial orientada apenas a um único produto — o SIAG-AP — sendo composta por pequenas equipas especializadas em várias áreas: desenvolvimento; administração de sistemas; implementação de projectos; administrativa/financeira e comercial. Apenas existem dois tipos de projectos dentro da organização, os de desenvolvimento focados na expansão de funcionalidades e melhorias da *framework* SIAG e os de implementação de projecto que surgem como um planeamento geral de integração dos projectos orientados a clientes, sendo estes concentrados no âmbito comercial, legislativo e contractual. Todas estas equipas respondem directamente aos dois elementos do quadro de administração, o Engenheiro Carlos Alves e o Engenheiro João Trindade.

2.2 Integração na Equipa de Desenvolvimento

A equipa de desenvolvimento na qual fui integrado cresceu no período inicial do meu estágio, sendo agora composta pelo chefe do projecto SIAG e co-orientador do meu estágio, Nuno Rodrigues, e, contanto comigo, oito programadores de java.

A minha integração foi feita através de várias formações dadas em conjunto com outros três elementos da equipa que entraram nos quadros da empresa por volta da mesma altura em que entrei. Nestas formações foi-nos introduzida a arquitectura e funcionamento actual do SIAG, assim como a nova arquitectura sobre o qual todos os futuros projectos serão desenvolvidos, a *framework* RAD[3] (*Rapid Application Development*[85]).

Capítulo 3

Planeamento

A proposta inicial de duração do projecto foi de nove meses a trabalhar a tempo inteiro.

3.1 Tarefas no Projecto

O projecto é composto pelas seguintes tarefas:

3.1.1 Planeamento e Acompanhamento do Projecto

Esta tarefa é caracterizada pelo acompanhamento feito ao longo de todo o projecto pelo chefe de projecto de forma a verificar o desenvolvimento e manter um planeamento adequado face aos problemas que possam surgir nas restantes tarefas.

3.1.2 Levantamento de Requisitos

Esta tarefa é composta por um período inicial de análise de mercado, onde foram investigados os conceitos e tecnologias actualmente usadas em aplicações desenvolvidas na área da BI, as tecnologias usadas na implementação do módulo de BI do SIAG-AP e ferramentas de ETL que possam vir a ser integradas no novo módulo. No final desta investigação houve uma fase de validação do âmbito do projecto feita pelo chefe de projecto e por um elemento da administração antes de proceder à próxima tarefa do projecto.

3.1.3 Análise de Requisitos

Esta tarefa é composta por pequenas iterações de suporte às diferentes fases de implementação, onde ocorre um período de análise e reavaliação das metodologias e técnicas utilizadas consoante as questões que foram surgindo à medida que a implementação do projecto progrediu.

3.1.4 Criação do Módulo SiagBI Independente

Esta tarefa é caracterizada pela criação do novo módulo SiagBI independente da *framework* SIAG. Para tal foi necessário retirar o código de BI actual para um módulo em separado e fazer com que este módulo comunique com o SIAG-AP (através do *Java Message System*[79]) de forma a não perder funcionalidades. Este processo é que permitiu preparar o projecto para a posterior inclusão das novas funcionalidades de ETL.

3.1.5 Independência de SiagBI das Fontes de Dados

Esta tarefa é caracterizada pelo estabelecimento da independência da camada de dados que suporta actualmente a *framework* SIAG. Todas as funcionalidades de BI presentes no SIAG-AP funcionavam sobre a sua camada de dados baseada em *Hibernate*[10] e pretendeu-se que as novas funcionalidades funcionassem independentemente desta camada, permitindo recolher informação de outras fontes de dados (que não provenham necessariamente do SIAG) de forma transparente ao utilizador. Numa fase final deste processo foi criada uma extensão ao modelo de dados do *SiagBI* de forma a suportar a definição de fontes de dados.

3.1.6 Implementação da Componente de ETL

Nesta tarefa foram implementadas as novas funcionalidades de ETL investigadas na fase inicial do projecto, conforme os requisitos funcionais aprovados pela organização dentro no novo módulo *SiagETL*.

3.1.7 Testes

No final de cada meta de desenvolvimento foi realizada uma fase de testes às funcionalidades do novo módulo *SiagBI* de maneira a garantir a sua qualidade e fiabilidade, assim como foram feitos testes sobre o módulo *SiagETL* à medida que foi desenvolvido.

3.1.8 Documentação para Utilizador Final

Nesta fase, foram criados os respectivos manuais de utilizador de forma a documentar as novas funcionalidades implementadas.

3.1.9 Criação de Relatório Preliminar

Esta tarefa é caracterizada pela criação de um relatório preliminar que serviu para reforçar o âmbito inicial do projecto.

3.1.10 Criação de Relatório Final

Esta tarefa é caracterizada pelo melhoramento do relatório preliminar, onde foi documentado todo o trabalho realizado até ao prazo de finalização do projecto estabelecido, assim como as suas conclusões finais e o trabalho futuro que poderá ser realizado na continuação do projecto.

3.2 Planeamento das Tarefas

3.2.1 Cronograma Inicial do Projecto

Seguidamente, apresenta-se o enquadramento cronológico final das tarefas descritas:

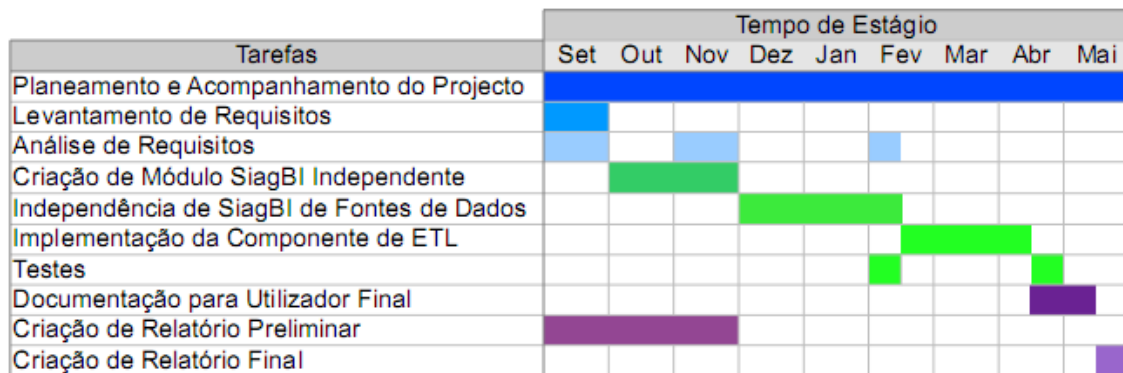


Figura 3.1: Cronograma do planeamento inicial do projecto

3.2.2 Cronograma Final do Projecto

Seguidamente, apresenta-se o enquadramento cronológico final das tarefas descritas:

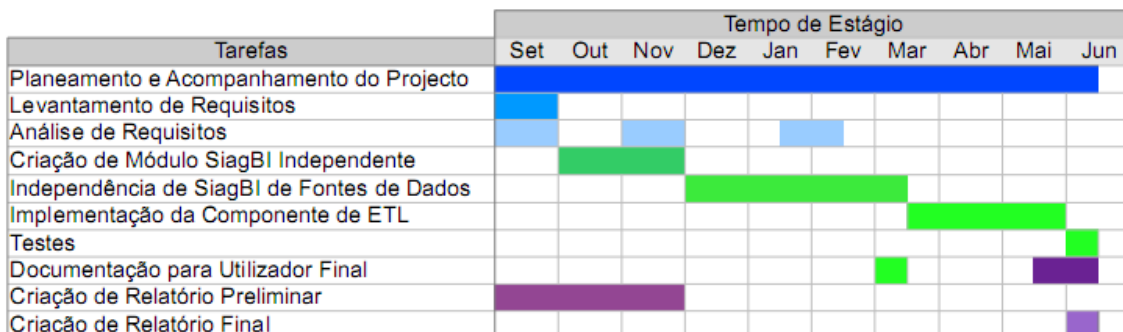


Figura 3.2: Cronograma do execução do projecto

3.3 Metodologia de Desenvolvimento

A metodologia de desenvolvimento adoptada baseou-se no modelo em cascata iterativo, orientado a funcionalidades. Para as diferentes tarefas deste modelo será utilizado um modelo de desenvolvimento ágil, com iterações de curtos períodos de tempo, sendo tipicamente menores que uma semana e podendo ir até quatro. No final destas iterações ocorre uma reunião com o gestor de projecto onde é feito um ponto de situação sobre o trabalho efectuado e onde são definidas as próximas iterações e possíveis correcções a fazer nas iterações concluídas até ao momento.

Esta proximidade com o estado de desenvolvimento do projecto permite estabelecer um melhor controlo sobre o que na realidade está a decorrer nas iterações definidas e poder reajustar as decisões tomadas face aos problemas encontrados.

Capítulo 4

Análise e Pesquisa

Neste capítulo irá ser exposta a informação recolhida na fase de análise do projecto e que serviu de base para o mesmo. Em primeiro lugar irá ser descrito alguns pontos importantes sobre o contexto do projecto, nomeadamente entender o que envolve um projecto de *Business Intelligence*[60] e de *Business Performance Management*[61]; perceber o que são *Data Warehouses*[72] e de que forma é que estão relacionados com as áreas de BI e de BPM, assim como os processos de *Extract, Transform and Load*[76] e outros conceitos relacionados. Estes aspectos estudados serviram para colocar em perspectiva como as ferramentas de ETL são usadas no mundo real e que requisitos funcionais mínimos é que são exigidos pelos gestores de negócio neste tipo de ferramentas. Por fim, irão ser referidos os requisitos definidos após a fase de pesquisa do projecto.

4.1 Business Intelligence

Business Intelligence[60] (BI) é um conceito que foi introduzido na década de 80 pelo *Gartner Group*[9] que refere toda a gama de tecnologias e mecânicas informáticas usadas para recolha, organização, análise, partilha e monitorização de informação necessária à gestão eficiente de um negócio. É um recurso importante no apoio às decisões de negócio, sendo tipicamente usado para catalogar informação de acordo com o seu historial, abrindo oportunidade a uma visão corrente ou até mesmo preditiva do sucesso de um negócio com base em certos factores.

Actualmente estes processos costumam ser agregados em pacotes de BI e de preferência integrados nos próprios sistemas de *Enterprise Resource Planning*[75] e de *Customer Relationship Management*[65] das empresas que adoptam estes novos métodos analíticos de contingência através de um ou mais módulos de *Business Performance Management*[61]. Os processos de BI tipicamente disponibilizam mecanismos de recolha e tratamento de dados, permitindo a construção de *data warehouses*[72], agregados lógicos de informação, que serão detalhados seguidamente neste capítulo. Após esta agregação de dados, segue-se para os processos analíticos de BI, nomeadamente os de criação de relatórios e os de

Online Analytical Processing[83], também discutidos mais à frente. Adicionalmente, alguns processos de BI também englobam métodos de *data mining*[70], vocacionados para a descoberta de nova informação por inferência do conhecimento da organização.

4.2 Business Performance Management

Business Performance Management[61] (BPM) é o conjunto de software e processos de análise e de gestão de informação que permitem a uma organização compreender os factores chave que contribuem para o sucesso de um negócio, contribuindo assim para a monitorização, análise e optimização da performance das organizações.

As soluções de BPM procuram resolver o problema da descentralização e elevado volume de dados dentro de uma empresa, procurando uma visão imediata do estado actual do negócio. É esta perspectiva em tempo real que possibilita uma acção constante face a um mundo de negócio sempre em evolução.

Para tal, uma solução de BPM tende a ser integrada nos próprio sistemas de ERP, recolhendo variadas métricas de performance (denominados de *Key Performance Indicators*[80]) e disponibilizando estas métricas em dashboards[66] desenhados pelos gestores de negócio, agregando métricas logicamente em função da área de negócio à qual dizem respeito. É esta flexibilidade que se procura numa solução de BPM, tornando-a adaptável face a qualquer contexto de negócio, abrindo portas para o sucesso de qualquer empresa baseada em sistemas de informação complexos.

Idealmente uma solução de BPM também permite afectar propositadamente os KPI's de forma a simular uma instabilidade do contexto sócio-económico da empresa e permitir compreender a eficácia da estratégia adoptada.

4.3 Data Warehouse

É um conceito que remonta à década de 80 quando Barry Devlin e Paul Murphy, ambos do departamento de investigação da IBM[35], definiram o termo de "*information warehouse*" e começaram a construir os primeiros *data warehouses*[72] (DW) experimentais. Trata-se de um sistema de armazenamento de dados que possui um modelo de dados simplificado que permite agregar informação dispersa dentro de uma organização, assim como uma elevada abstracção dos dados e da sua implementação. Estes dados podem ser provenientes de várias fontes de dados diferentes dentro da empresa (tipicamente consolidadas através de ferramentas de ETL[76]), independentes da sua implementação e que dão suporte a operações de *business intelligence*, tais como operações OLAP[83], operações de *reporting*, assim como variadas tarefas de análise. A informação contida num DW é a soma de vários sub-conjuntos de dados divididos por várias unidades lógicas menores denominadas **data marts**[69]. Os **data marts** são representações mais especializadas de

todo o espectro de informação contido no *data warehouse* e que tipicamente são criados para compreender uma única área de negócio. A criação de *data warehouses* é um dos primeiros passos adoptados para conseguir agregar a informação que se acha relevante face ao contexto de negócio, simplificando o acesso à informação.

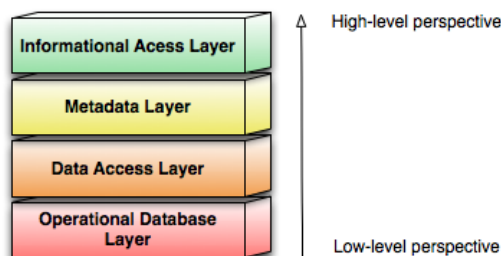


Figura 4.1: Camadas da arquitectura de um Data Warehouse

4.3.1 Arquitectura

A arquitectura de um *data warehouse* é composta por quatro camadas, tal como é descrito na figura 4.1: *Informational Access Layer*, *Metadata Layer*, *Data Access Layer* e *Operational Database Layer*.

Na **Operational Database Layer** é onde se situa a origem dos dados. É nesta camada mais baixo-nível que se situam os sistemas ERP e CRM das empresas sobre o qual o *data warehouse* irá actuar.

Seguidamente pode-se encontrar a **Data Access Layer** que serve de interface entre a *Operational Database Layer* e a *Metadata Layer*, é nesta camada que se encontram as ferramentas de *Extract, Transform and Load*[76] que irão processar os dados e prepará-los para futuras tarefas analíticas que irão actuar sobre o DW.

Após a **Data Access Layer** tem-se a *Metadata Layer*, que se pode designar como sendo o dicionário de dados[68] do DW. Esta camada serve de repositório centralizado de informação sobre os tipos de dados, estrutura, relações, origem e formato presentes no sistema e que actua como camada de suporte, reforçando uma estandardização dos dados presentes no DW e contribuindo para um conhecimento organizacional coerente.

Por fim, existe a **Informational Access Layer**, a camada de mais alto nível do DW sobre a qual se executam os processos de reporte e de OLAP referidos anteriormente.

4.3.2 Modelo de Dados

O design de um *data warehouse* tem sido um dos temas mais controversos desde o seu aparecimento. Para construir um DW, tem de se tomar duas decisões de arquitectura

fundamentais: Como armazenar a informação em cada componente da sua estrutura e como modelar essa mesma estrutura para responder às necessidades do negócio.

Dentro da problemática do armazenamento de dados, existem duas opções:

- Armazenar os dados num **modelo dimensional**
- Armazenar os dados num **modelo relacional normalizado**

Num **modelo dimensional**, os dados são particionados em *factos* (ou *medidas*) onde são armazenados os dados numéricos de um registo da base de dados, ou *dimensões* onde são guardados todos os dados complementares dessa mesma informação numérica (por exemplo, nomes, códigos, etc.). Este modelo obedece tipicamente a um *esquema em estrela* onde se tem uma única tabela de factos central relacionada à distância de um nó com as várias dimensões de suporte, como se pode verificar no exemplo da figura 4.2. Pode também ser incorporado num esquema *floco de neve* onde eventualmente uma ou mais dimensões possuem ligações para uma outra "sub-dimensão".

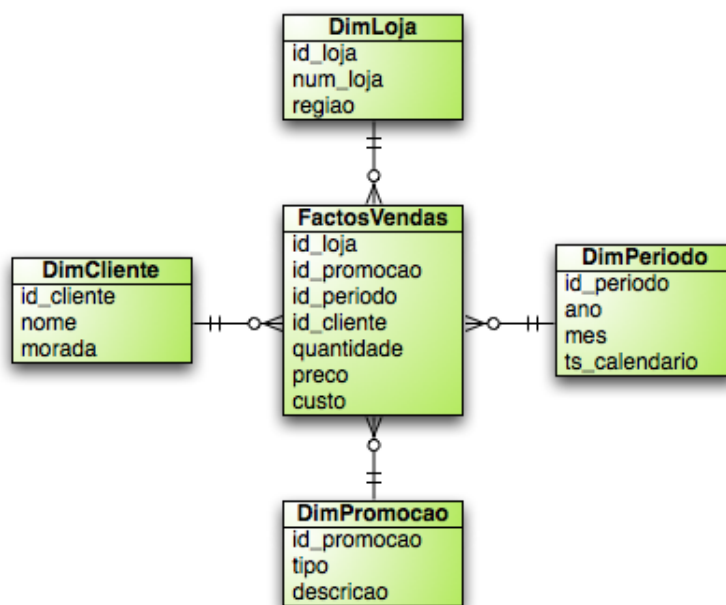


Figura 4.2: Exemplo de um esquema em estrela

A maior vantagem na utilização deste tipo de modelos passa pela forma simplificada de como é organizada a informação. Cada registo no *data warehouse* está organizado em medidas de um ou mais processos de negócio (por exemplo, valor líquido de vendas de um departamento, número de pessoas contratadas pelos recursos humanos, etc.) cruzados com a informação contida em vários contextos dimensionais (por exemplo, data de aquisição ou cliente associado a uma venda, departamento a que um empregado pertence, etc.) sendo muito mais compreensível na perspectiva dos utilizadores de negócio, público alvo principal das aplicações de BI e de BPM.

Esta estrutura de armazenamento de dados também costuma ser tipicamente mais eficiente na busca de dados em operações de OLAP[83], visto que o número de tabelas envolvidas é reduzido (em comparação com outros modelos de dados mais complexos, como é o caso do modelo relacional normalizado) e a distância reduzida entre as tabelas facilita a agregação ágil de dados por parte das operações de OLAP[83].

No entanto, a adoção deste modelo possui algumas desvantagens, onde se pode realçar a falta de adaptabilidade da sua estrutura face a mudanças dos processos de negócio suportados pelo *data warehouse*. Outra desvantagem, embora tenha vindo a diminuir face à evolução rápida das ferramentas de ETL[76], é a manutenção da integridade dos dados num *data warehouse* quando são recolhidos de vários sistemas transacionais diferentes, cada um com as suas regras e sintaxe próprias de armazenamento de dados.

Para concluir, ambos os modelos abordados também podem ser implementados em modelos híbridos, embora essas abordagens não seja analisadas, uma vez que não caem no âmbito deste documento.

Num **modelo relacional normalizado** os dados são divididos em entidades (por exemplo, cliente, venda, produto, departamento, empregado, etc.), cada uma destas representando uma tabela na base de dados, com diversas relações entre si, sendo depois agrupadas em diversas categorias (continuando o exemplo anterior, agrupar as entidades referidas num grupo para atender todos os pedidos de vendas de uma empresa por parte da aplicação em questão).

O resultado desta abordagem é um conjunto grande de tabelas com várias relações entre si, à semelhança da maioria das camadas de persistência de aplicações empresariais actuais.

Pode-se considerar que este modelo é bastante mais fácil na adição de novos registos num *data warehouse*, tendo em conta que cada tabela é auto-explicativa relativamente ao seu contexto num processo de negócio. No entanto, a sua estrutura complexa torna difícil agrupar os dados de forma significativa, assim como aceder à informação armazenada sem uma compreensão detalhada da sua estrutura e das diversas fontes de dados envolvidas.

4.3.3 Princípios de Design

Após a escolha do modo de armazenamento dos dados, há que estruturar o *data warehouse* de acordo com as necessidades de negócio.

Pode-se adoptar várias abordagens distintas face a este problema:

- A abordagem **bottom-up**
- A abordagem **top-down**
- A abordagem **híbrida**

Na abordagem **bottom-up**, defendida por um dos pioneiros no mundo do *data warehousing*, Ralph Kimball[95], começa-se por construir vários pequenos *data marts*[69] modelados tipicamente para atender a diferentes áreas dos processos de negócio. Estas construções são depois integradas no que é chamado por Kimball de *Data Warehouse Bus Architecture*[16]. Nesta arquitectura, dimensões que são partilhadas entre diversos *data marts*[69] sofrem um processo de normalização de dados para que se possa controlar a redundância dos dados presentes no *data warehouse*. Apesar desta metodologia se apelidar de **bottom-up**, dado os requisitos que foram especificados anteriormente sobre a redundância de dados, a construção dos primeiros *data marts* é apenas feita após uma análise inicial **top-down** orientada ao negócio.

Pode-se verificar então que cada *data mart* pertencente a este modelo é um módulo independente dos restantes componentes da estrutura do *data warehouse*, embora seja sempre dependente do barramento (*bus*) sobre o qual foi implementado (tem de obedecer às regras de normalização de dados impostas nas dimensões partilhadas das quais usufrui presentes no barramento). Os dados presentes nos diferentes *data marts* podem depois ser cruzados através das referências comuns para as dimensões partilhadas no barramento, sendo esta operação de análise designada como uma operação de *drill-across*, em contraste com as operações de *drill-through* que apenas cruzam dados presentes num único *data mart*.

A maior vantagem da implementação desta arquitectura passa pelo retorno rápido de investimento, uma vez que os primeiros *data marts* surgem rapidamente para gerar os primeiros relatórios. A integração de novos *data marts* complementares é feita de forma ágil, necessitando apenas de respeitar a formatação dos dados presentes nas dimensões do *bus*, podendo ser facilmente integrados sem um planeamento extenso de toda a estrutura dos dados do *data warehouse*. Os diversos *data marts* podem também estar distribuídos por diversos servidores, permitindo uma maior escalabilidade nos sistemas dos quais fazem parte.

Na abordagem **top-down**, defendida também por outro dos pioneiros do mundo do *data warehousing*, Bill Inmon[58], existe apenas um *data warehouse* centralizado onde são guardados todos os dados atómicos transacionais (equivalentes aos factos impostos pelo modelo dimensional de Kimball), para depois poderem ser integrados num modelo de dados relacional normalizado como descrito anteriormente. Os dados complementares aos nossos dados atómicos transacionais recebidos pelo *data warehouse* são depois repartidos em diversos *data marts* especializados em diferentes áreas dos processos de negócio. O resultado é um *data warehouse* centralizado com dimensões de elevada consistência e coerência uma vez que os *data marts* são sempre gerados através da informação contida no repositório central.

A maior vantagem na utilização desta metodologia é a elevada flexibilidade permitida na estrutura adoptada pelas fontes de dados empresariais sobre as quais o *data warehouse*

actua. Uma mudança no modelo de dados resulta num impacto mínimo sobre a estrutura do DW, uma vez que apenas se torna necessário criar um novo *data mart* que possa responder a esta mudança. Existem também outras vantagens, tais como: a minimização de *data marts* renegados no sistema, dado a sua dependência do DW central; a elevada consistência e estandarização sobre os dados do sistema e a redução da redundância dos dados que daí advém.

No entanto, é de notar que esta abordagem requer um planeamento inicial profundo, tendo em conta a complexidade e âmbito vasto dum projecto deste tipo, resultando num gasto de recursos inicial considerável, assim como um retorno lento do investimento inicial até se poder ter os primeiros *data marts* em funcionamento.

Na abordagem **híbrida** procura-se obter no *data warehouse* final a agilidade e simplicidade da implementação presente na abordagem *bottom-up*, sem sacrificar a vantagem da integração global da abordagem *top-down*. Para tal são construídos dois *data warehouses* integrados, sendo cada um construído com uma metodologia diferente (um em *bottom-up* e outro em *top-down*. A ideia principal por detrás desta arquitectura passa por armazenar todos os dados atómicos no DW com arquitectura *bottom-up*, passado gradualmente estes dados para o DW com arquitectura *top-down* à medida que vão sendo necessários para operações de BI. Desta forma é reforçada a gestão, integração e exploração global de dados para toda a organização.

4.4 Extract, Transform and Load

Extract, Transform and Load[76] (ETL) é um processo de base de dados, essencial no sistema que compõe um *data warehouse*, responsável por popular as suas tabelas com os dados de negócio necessários. Tal como o seu nome indica, é um processo dividido em três fases:

- A fase inicial de extracção de dados — **Extract**
- A fase intermédia de transformação dos dados — **Transform**
- A fase final de carregamento de dados — **Load**

A fase de **extração** de dados é caracterizada pela transparência na recolha de dados de sistemas diferentes, entre os quais sistemas de base de dados relacionais e não-relacionais, *flat files*, etc. Nesta fase também podem eventualmente ocorrer processos de *data quality*[71] de forma a rejeitar alguns dados que não obedeçam às regras de negócio impostas no *data warehouse* (por exemplo, não aceitar dados relativos a vendas relacionadas com um determinado fornecedor).

Seguidamente, na fase de **transformação** de dados, são aplicadas um conjunto de regras e funções sobre os dados de forma a que se reforcem determinados padrões sintáticos

em todos os dados que compõe o universo do *data warehouse*. Desta forma é possível certificar de que toda a informação no DW é coerente, independentemente de ter sido recolhida de sistemas de informação diferentes, cada um obedecendo aos seus próprios padrões sintáticos. Em adição, podem também ser aplicados em processos de qualidade de dados de forma a ignorar certos registos da fonte de dados em questão. Eis algumas operações exemplares que costumam ocorrer nesta fase:

- Ignorar registos que tenham um determinado campo nulo.
- Extrair apenas certas colunas de um registo duma base de dados.
- Traduzir certos valores que na fonte de dados se encontravam codificadas.
- Codificar um determinado campo de um registo.
- Aplicar formulas em dados numéricos.
- Gerar novas colunas nos registos derivadas da informação presente noutros campos.
- Ordenação de registos.
- Agregação de registos.
- Aplicar validações sobre registos.

Por fim, após nossa disposição os dados relevantes para o *data warehouse*, já normalizados e tratados de acordo com as regras de negócio, ocorre a fase de **carregamento** dos dados para o *data warehouse*. Existem várias opções sobre a forma de como popular o DW. Dependendo das regras de negócio impostas, um DW ao receber novos dados pode sobrescrever dados já existentes cumulativamente. Pode também, considerando o DW um histórico de tudo o que se passa na organização, acrescentar os novos dados sem nunca apagar registos já existentes (podendo tornar-se numa ferramenta importante de auditoria).

4.5 Online Analytical Processing

Online Analytical Processing[83] (OLAP) é uma abordagem que permite analisar grandes volumes de dados de forma interactiva, simples e ágil, sendo a metodologia principal de análise de dados em aplicações de BI. Bases de dados que suportem operações de OLAP possuem um modelo de dados dimensional[16], permitindo a realização de consultas (*queries*) analíticas complexas e ad-hoc sobre múltiplas dimensões, assegurando sempre uma boa performance.

Tudo começa na criação de estruturas de dados designadas de *cubos OLAP*[64]. Os *cubos* são estruturas de dados simples que obedecem ao modelo dimensional de dados

descrito anteriormente neste capítulo. São compostas por uma tabela de factos (onde são armazenados os dados numéricos relevantes) e um conjunto de dimensões associadas a esses factos (onde é armazenada informação complementar sobre esses dados numéricos). Pode-se dizer que a tabela de factos é um conjunto de *medidas* sobre um determinado factor e que essas medidas contêm um conjunto de *etiquetas*. As dimensões são o que traduzem essas mesmas *etiquetas* para a informação definitiva sobre um determinado facto.

Após o *cubo OLAP* ser definido para um sub-conjunto do universo que compõe o *data warehouse*, pode-se efectuar um conjunto de consultas (*queries*) sobre o cubo para cruzar a informação contida na tabela de factos e as dimensões. O resultado é uma matriz de valores onde os eixos das colunas e linhas representam uma ou mais dimensões e as *medidas* são os valores que a preenchem.

Existem vários tipos de sistemas OLAP, entre os mais polulares:

- **MOLAP** — *Multidimensional Online Analytical Processing*[82]
- **ROLAP** — *Relational Online Analytical Processing*[86]
- **HOLAP** — *Hybrid Online Analytical Processing*[77]

O **MOLAP** é o modo clássico de OLAP, onde os dados são pré-processados para que possam ser armazenados no cubo antes de serem retornados sobre a forma de resultados para o utilizador. Esta metodologia permite um armazenamento especializado feitos através de indexação multi-dimensional e *caching*. A grande vantagem deste tipo de sistemas passa pelo tempo reduzido em que são feitas *queries* sobre o sistema, embora o pré-processamento dos dados implique uma sincronização mais lenta com a fonte de dados em questão.

Em alternativa, existe o **ROLAP** que, em contraste com o *MOLAP*, não necessita de processar previamente os dados. É feita uma inspecção aos dados presentes numa base de dados relacional e gerada uma *query* de SQL apropriada ao nível do pedido realizado pelo utilizador. A falta de pré-processamento implica uma sincronização mais rápida com a fonte de dados em questão, embora as *queries* sobre o sistema sejam de execução mais lenta comparativamente a sistemas *MOLAP*.

No caso do **HOLAP** trata-se de uma mistura de uso das tecnologias de *MOLAP* e de *ROLAP*, embora não exista uma definição específica de como é efectuada essa mesma combinação.

4.6 Levantamento de Requisitos

Nesta secção irá ser descrito o porquê da inexistência de requisitos funcionais para o projecto e descrever os requisitos não-funcionais definidos pelo projecto.

4.6.1 Requisitos Funcionais

Neste projecto não foram definidos requisitos funcionais uma vez que se trata de um projecto que procurou a inclusão de novas tecnologias de forma inovadora, sem estar limitado pela definição *a priori* de requisitos funcionais. Dado este panorama, não houve necessidade pela parte da administração da GEDI da especificação formal de requisitos funcionais.

Foi dada a liberdade total na escolha de funcionalidades face ao que foi aprendido na fase de estudo de mercado e com base na implementação escolhida, sendo apenas exigido a inclusão de uma ferramenta de ETL à escolha da equipa de desenvolvimento que fosse apelativa a gestores de negócio e que fosse previamente aprovada pela administração em termos de usabilidade.

4.6.2 Requisitos Não-Funcionais

Os requisitos não-funcionais aqui apresentados na tabela 4.1 foram especificados iterativamente ao longo de reuniões semanais de desenvolvimento.

Código	Requisito
rnf-1	Todas as funcionalidades de BI implementadas até à data devem passar para o servidor baseado em comunicação por JMS[79] que funciona independentemente do SIAG — o <i>SiagBI</i> .
rnf-2	As novas funcionalidades de ETL deverão ser implementadas num outro servidor baseado em comunicação por JMS[79] independente do SIAG — o <i>SiagETL</i> .
rnf-3	Os módulos <i>SiagBI</i> e <i>SiagETL</i> terão de ser baseados na <i>framework</i> de comunicação por JMS[79] já existente e idealmente ser integrados na lógica já implementada no modelo <i>Model-View-Controller</i> [13] do SIAG.
rnf-4	A camada de persistência de ambos os módulos deverá continuar do lado do SIAG e cabe aos servidores apenas atender pedidos específicos de BI e ETL da camada de negócio.
rnf-5	Os pedidos feitos ao SIAG têm de ser independentes da sua camada de persistência já existente, não podendo conter objectos gerados a partir do <i>Hibernate</i> [10].
rnf-6	As bibliotecas utilizadas pelo <i>SiagBI</i> do <i>Mondrian</i> e do <i>JPivot</i> devem ser actualizadas para a última versão, na condição de não perder qualquer funcionalidade já implementada.
rnf-7	Os pedidos feitos a ambos os servidores devem ter em conta a resolução de problemas comuns de concorrência.
rnf-8	A interface de criação de cubos OLAP <i>Mondrian</i> [31] já existente deve ser refeita sobre a nova plataforma de interfaces do SIAG — o <i>SiagGWT</i> [3].
rnf-9	Todas as novas interfaces respectivas do novo módulo de ETL terão de ser implementadas sobre a plataforma <i>SiagGWT</i> .
rnf-10	As transformações devem poder ser agendadas consoante um calendarização especificada pelo utilizador.
rnf-11	O agendador de transformações de ETL deve ser implementado sobre a plataforma de <i>Business Process Management</i> já existente no SIAG - o <i>SiagJBPM</i> .

Tabela 4.1: Tabela de requisitos não-funcionais do projecto.

Capítulo 5

Estudo de Mercado

Neste capítulo irá ser exposta a análise que foi feita sobre o *Pentaho Business Intelligence Suite*[21], um pacote de software com variadas funcionalidades de BPM, de forma a tentar delinear factores que são decisivos na popularidade das soluções de BPM actuais, assim como aspectos funcionais que pudessem ser incluídos no projecto desenvolvido. Para tal, irá ser dado um foco especial sobre o *Pentaho Business Intelligence Server*, o servidor que orquestra todas as actividades de BPM no pacote da *Pentaho*, e o seu componente de extracção, transformação e carregamento de dados, o *Pentaho Data Integration*, também conhecido por *Kettle*.

Por fim, irá ser feita uma comparação do *Pentaho Data Integration* em termos de funcionalidades e outros critérios detalhados na secção 5.2 com algumas alternativas *open-source* presentes no mercado de ferramentas de ETL.

5.1 Pentaho Business Intelligence Suite

A *Pentaho BI Suite*[21] é um conjunto de ferramentas comerciais produzidas pela empresa *Pentaho*[36] que no seu todo visam atender variados requisitos de BI/BPM de forma simplificada.

Este pacote de *software* é distribuído em duas versões distintas, uma versão de código aberto disponibilizado pelo repositório da *SourceForge*[37] e uma versão profissional paga que contém algumas funcionalidades e componentes extra exclusivos desta versão (como é o caso da *Pentaho Administration Console*[26]), assim como suporte ao cliente. Entre as suas funcionalidades, pode-se encontrar funcionalidades de OLAP[83], ETL[76], *reporting*, criação e gestão de *dashboards*[66], *data mining*[70], análise interactiva, etc. pode-se ver mais em detalhe como estas funcionalidades são agrupadas de uma perspectiva alto nível através da figura 5.2.

O pacote da *Pentaho* distingue-se no mercado principalmente devido à sua filosofia de *Agile BI*[27]. Esta perspectiva surge como resposta aos problemas comuns de grande parte das ferramentas de BI actuais: elevada complexidade de implementação de soluções

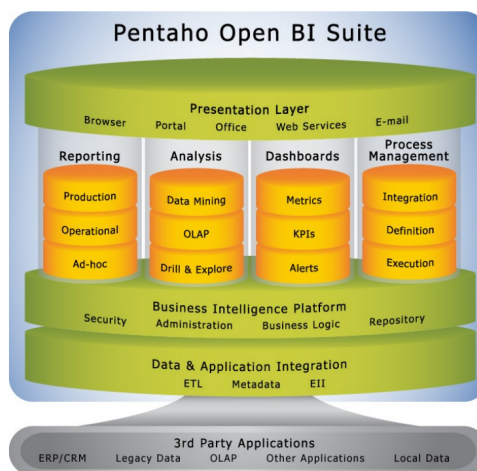


Figura 5.1: Modelo alto-nível dos componentes que constituem a *Pentaho Business Intelligence Suite*

de BI; falta de flexibilidade nas soluções criadas face a novos paradigmas de negócio; falta de uma visão integrada entre desenho, modelação e visualização de soluções de BI e dos seus resultados.

Para atingir estes objectivos, todas as actividades que se executam no pacote da *Pentaho* são baseadas em processos de negócio definidos pelos próprios utilizadores. Desta forma, existe à partida uma redução significativa da complexidade de implementação de soluções de BI, uma vez que cada solução pode ser mapeada a um ou mais processos especificados no sistema, que em si pode despoletar a execução de outros sub-processos. Cada processo pode também implementar alguma lógica de negócio, como por exemplo na atribuição de tarefas a utilizadores específicos, responsáveis pela execução de um determinado segmento do processo. O resultado é uma plataforma compacta, organizada e flexível de fluxo de trabalho entre utilizadores e ferramentas, orientada a processos / criação de soluções.

Em termos de componentes, este pacote de *software* é centrado à volta do *Pentaho Business Intelligence Server*, responsável por todo o mecanismo de gestão de processos e actividades, contendo depois algumas ferramentas de cliente para interagir com este servidor (à excepção do *Pentaho Administration Console*, uma ferramenta de monitorização do servidor). Estes componentes são:

- **Pentaho Business Intelligence Server** — Servidor responsável por toda a actividade de BI que irá ser descrito seguidamente em mais detalhe neste capítulo.
- **Pentaho Administration Console** — Uma consola de administração do servidor de BI disponível apenas na versão paga da *suite*.
- **Pentaho Design Studio** — Um editor gráfico de processos de BI, produzindo ficheiros XML denominados de *Action Sequences* para posteriormente serem envia-

dos e interpretados pelo servidor. É nesta ferramenta que são especificadas a ordem e fluxo de acções de BI (extração de dados de um ficheiro local, criação de reports, notificação de utilizadores por e-mail, criação de sub-processos, etc.) dentro de um determinado processo.

- **Pentaho Metadata Editor** — Um editor gráfico de meta-dados. O servidor da *Pentaho* de forma a reforçar o uso de determinados formatos e comportamentos consistentes nos dados que passam para o motor de *reporting*, recorre a uma camada de meta-dados onde estão definidas estas regras. Este editor permite assim associar estas regras a determinadas tabelas ou campos da base de dados, permitindo uma abstracção da implementação física dos dados.
- **Pentaho Report Designer** — Um editor gráfico de relatórios, num modo de construção de relatórios semelhante ao editor gráfico da *Crystal Reports*[44]. Os relatórios são organizados em grupos ou "listras" onde são atribuídos determinados sub-conjuntos de dados que constituem o relatório. Estes dados podem ter sido previamente tratados pela camada de meta-dados, através de regras de sintaxe e formatação definidos pelo *Pentaho Metadata Editor*.
- **Pentaho Schema Workbench** — Um editor gráfico de *schemas* para o *Mondrian*[31], o servidor OLAP da *Pentaho*. O *Mondrian* necessita destes ficheiros XML para a definição do modelo de dados sobre o qual irá actuar nas explorações de dados dos utilizadores. Esta tecnologia irá ser abordada mais tarde, uma vez que já se encontra integrada no SIAG[33].
- **Pentaho Aggregation Designer** — Um editor gráfico de tabelas de agregação. O servidor *Mondrian*, quando se trata de um grande volume de dados, por vezes recorre a estas tabelas como forma de melhoria da sua performance em explorações de dados OLAP. Estas tabelas tratam-se apenas de pré-processamentos de uma combinação possível de dimensões com os factos dentro de um cubo OLAP[64]. Desta maneira evita-se a repetição de um processamento extenso feito apenas no momento em que um utilizador explora uma determinada combinação de dimensões com uma determinada perspectiva. Os dados em vez de serem explorados e processados apenas no momento em que são requisitados pelo utilizados, são extraídos directamente da tabela de agregação correspondente a essa perspectiva, havendo uma poupança significativa de tempo na exploração de dados.
- **Pentaho Analyser** — Um dos componentes mais característicos dentro da *suite*. Trata-se de uma interface *web* intuitiva que permite a exploração de cubos OLAP, criação de *dashboards*[66] e criação de relatórios básicos de forma rápida e simplificada.

- **Pentaho Data Integration** — O componente responsável pelos processamentos de ETL[76] desta plataforma que irá ser abordado em mais detalhe na secção 5.2 de análise de ferramentas ETL.

Para esta análise, irá ser feito um foco especial sobre o *Pentaho Business Intelligence Server* pela sua importância no modo de funcionamento da plataforma no seu todo e sobre o *Pentaho Data Integration* por ser a ferramenta responsável pelos processos de ETL[76] do pacote da *Pentaho*.

5.1.1 Pentaho Business Intelligence Server

O *Pentaho BI Server* é o componente central da *Pentaho BI Suite* responsável por toda a gestão e execução das actividades de BI. É um servidor desenvolvido na linguagem *java* que executa sobre o *JBoss Application Server*[15], embora possa correr sobre qualquer servidor web compatível com J2EE. É constituído por um motor central de *workflow*[89], o *Enhydra Shark*[51] aquiescente com as normas de *Business Process Management Notation*[46] e de *XML Process Definition Language*[90] como standards de definição de processos de negócio. O motor interpreta documentos XML denominados de *Solution Definition Documents* que contêm:

- A definição dos processos de negócio.
- As definições das actividades que executam arbitrariamente como parte dos processos (definições de fontes de dados, *queries* a executar, *templates* de relatórios, regras de entrega e notificação de utilizadores, etc.).
- As relações entre os elementos descritos nos pontos anteriores.

São estes ficheiros que ao serem introduzidos no servidor despoletam as variadas actividades de BI definidas para executar no sistema, seja através de eventos lançados pelo motor de *workflow* ou de eventos lançados pelo motor de agendamento de tarefas (permitindo activar as actividades consoante um intervalo de tempo definido ou um calendário de execuções).

O servidor foi concebido de forma a dar suporte à integração com aplicações externas — *Enterprise Application Integration*[74] — de maneira a facilitar o seu uso integrado e em paralelo com outras aplicações empresariais. Na figura 5.2 pode-se verificar que o acesso feito aos diferentes componentes de BI (também designados por motores) é todo feito a partir da camada designada no esquema por *Solution Engine*, o motor de soluções. Os diversos serviços do servidor disponibilizam suporte a *web services*, abrindo oportunidade à integração com aplicações externas que interpretem os dados devolvidos por estes. Para além de *web services*, a plataforma também possibilita a integração com outros sistemas através da definição de fontes de dados externas, permitindo assim a criação

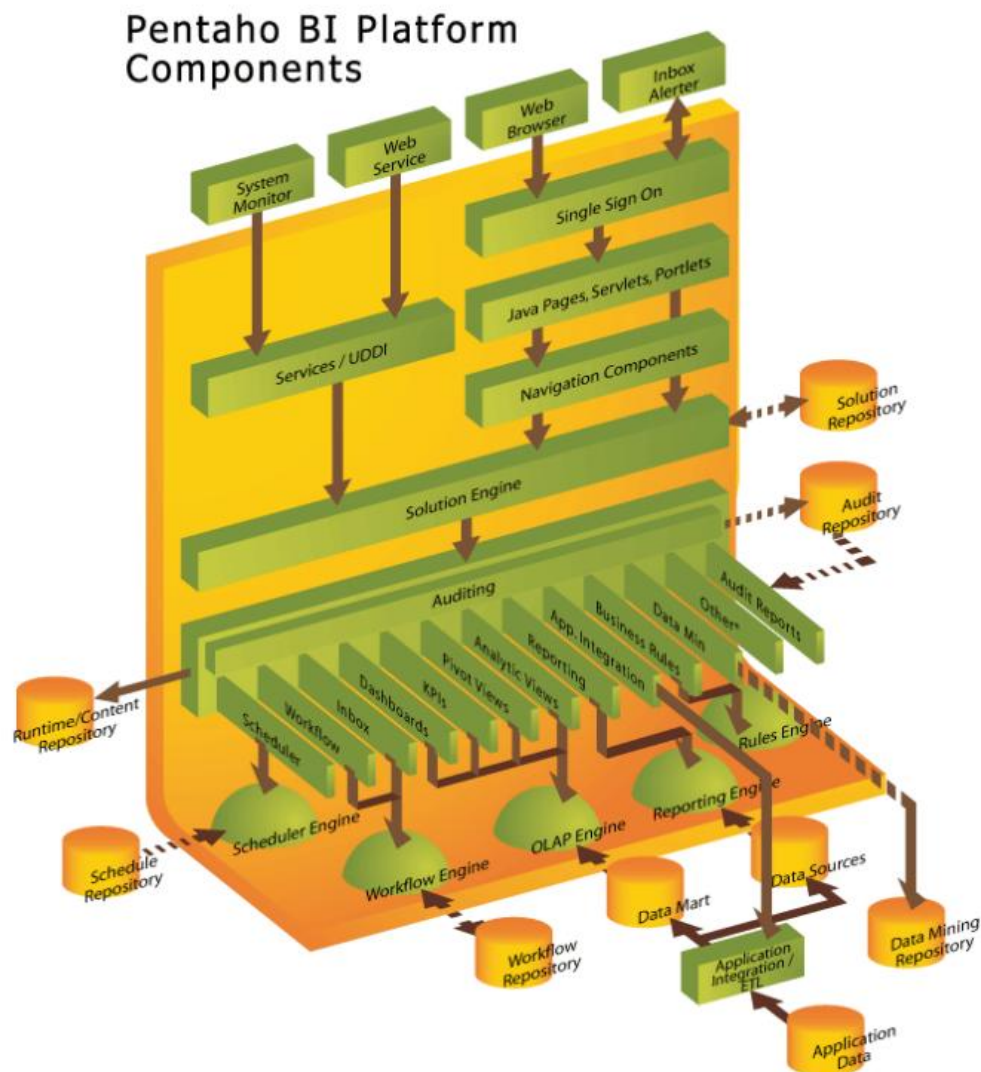


Figura 5.2: Modelo alto-nível dos componentes que constituem o *Pentaho BI Server*

de relatórios, cubos OLAP, *data warehouses*, etc. que podem actuar sobre dados que não pertencem à aplicação em si.

A informação possível de obter por um utilizador depende sempre do repositório presente no motor de *workflow* de forma a garantir segurança no acesso a dados restritos. Neste repositório pode-se encontrar um histórico de toda a actividade relativa a processos que passa pelo servidor, tornando possível fazer auditorias (duma perspectiva histórica ou até mesmo em tempo real) sobre a informação que flui entre processos no sistema. Esta capacidade de auditoria não está apenas limitada a informação relativa processos de *Business Process Management*[61], uma vez que todos os componentes que constituem os diferentes motores do servidor (motor de *reporting*, OLAP, ETL, etc.) estão directamente ligados a uma camada de auditoria, sendo assim possível validar a actividade de

um utilizador com um determinado componente.

O servidor possui uma infra-estrutura que providencia mecanismos de administração de sistema avançados, entre os quais serviços de monitorização de sistema (através de *Simple Network Management Protocol*[87]), relatórios de utilização, ferramentas de validação de configuração e ferramentas de diagnóstico. Possui também componentes que permitem a análise e criação de relatórios de performance de processos de BI. Estas ferramentas permitem realizar operações analíticas de *slice-and-dice*[57], *what-if* e *datamining*[70] sobre os artefactos relacionados com o *workflow* de um determinado processo presente no sistema.

Dentro da plataforma existem também variados motores de regras para que a lógica de negócio seja organizada entre diversos motores diferentes, esteja exposta e acessível aos utilizadores e seja facilmente customizável pelos mesmos. Existem também alguns componentes não representados na figura 5.2 tais como os componentes de impressão, e-mail, formatação de mensagens, gestão de atributos de instâncias de processos BPM[61], relatórios de performance, etc.

Cada motor presente no sistema (representados na figura 5.2 abaixo da camada do motor de soluções) possui um componente responsável pela sua integração na plataforma. Isto permite uma elevada flexibilidade, na medida em que a arquitectura está preparada para uma eventual troca de motores ou até mesmo a adição de novos motores dado que os componentes necessários para a sua integração são criados.

Em termos de tecnologias e ferramentas que estejam a ser utilizadas pela *suite* da *Pentaho*, pode-se referir:

- **JBoss Application Server**[15] — Utilizado como servidor web compatível com J2EE.
- **JBoss Portal**[15] — Uma plataforma de suporte a *portlets*.
- **Java Server Pages**[20] (JSP), **Servlets** e **Portlets** — Utilizadas nas interfaces web.
- **Hibernate**[10] — Uma *framework java* de persistência.
- **Mondrian OLAP Server** — O servidor de OLAP utilizado, concebido pela própria *Pentaho*.
- **jPivot Analysis Front-end** — Bibliotecas JSP que dão suporte a exploração de dados em servidores OLAP, disponibilizando um API para realizar operações de *slice-and-dice*[57], *drill-down*[73] e *roll-up* sobre os dados, usando o *Mondrian* como servidor OLAP.
- **Firebird RDBMS**[7] — O servidor de base de dados utilizado para todos os repositórios internos no servidor.

- **Enhydra Java Workflow Editor**[51] — Um editor gráfico de *workflow* implementado sobre *Swing*[88].
- **Enhydra Shark**[51] — O servidor de *workflow* que suporta os ficheiros criado pelo *Enhydra Java Workflow Editor*.
- **Pentaho Data Integration** (referido na secção 5.2.1) — O componente responsável por executar operações de ETL[76] e de *data warehousing* na plataforma, fazendo parte do pacote do *Pentaho Data Integration*[47].
- **Weka Data Mining**[24] — Uma ferramenta que disponibiliza algoritmos inteligentes de *data mining*[70].
- **Eclipse Workbench** e componentes **BIRT**[45] — Um sistema de criação de relatórios baseado nos componentes gráficos do *eclipse*.
- **Java Open Single Sign-on**[48] — Um componente *open-source* especializado na autenticação *single sign-on* que providencia também suporte a autenticações por *Lightweight Directory Access Protocol*.
- **Mozilla Rhino Javascript Processor**[18] — Um interpretador de *javascript* acessível a partir a *Java Virtual Machine* (JVM).

5.2 Ferramentas de Extract, Transform and Load

Nesta secção irão ser mostradas as ferramentas *open-source* de ETL que foram estudadas para a possível integração com o SIAG[33], fazendo uma comparação entre os diversos produtos. Estas ferramentas foram escolhidas tendo em conta vários factores, entre os quais: preferência por projectos de código aberto; nível de popularidade na *web*; maturidade das empresas por detrás das ferramentas; disponibilidade de documentação; usabilidade do produto com foco a utilizadores da área de negócio. Para a comparação propriamente dita foi elaborado um pequeno exemplo de teste (criação de um *data warehouse* básico com algumas transformações de dados) com o objectivo de analisar também a curva de aprendizagem do utilizador e a simplicidade de uso. Este documento mostra apenas os aspectos conclusivos da análise feita, não abordando o exemplo e as implementações em si. Para obter mais detalhes a nível de implementação do exemplo nas diversas ferramentas pode-se recorrer ao documento em anexo *Análise de Dados (PEI)*.

5.2.1 Pentaho Data Integration

O *Pentaho Data Integration*, também conhecido por *Kettle* (*Kettle Extraction, Transport, Transformation and Loading Environment*), é o conjunto de ferramentas *open-source* res-

ponsável pelos processos de ETL da *Pentaho Business Intelligence Suite*[21]. A sua característica principal é ser baseado em modelos (guardados sobre a forma de meta-dados) que representam as transformações e os fluxos de dados que ocorrem num determinado processo de ETL. Serve assim de intermediário entre as diversas fontes de dados que constituem o conhecimento disperso de uma organização com o objectivo de conseguir construir *data warehouses* consoante determinadas regras de negócio que se quer impor sobre os dados. Estes dados irão depois ser ponto de partida para outras actividades de *Business Intelligence* como pode-se observar na figura 5.3.

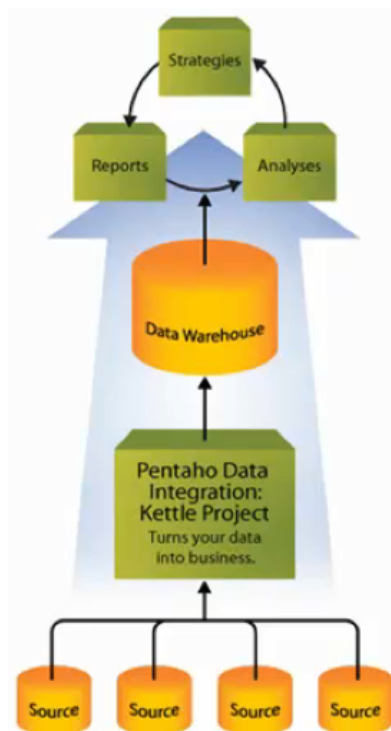


Figura 5.3: Esquema do funcionamento do Pentaho Data Integration

Observando a figura 5.4, pode-se verificar que os modelos dos processos de ETL são compostos por dois artefactos diferentes: as *transformations* (transformações) que são usadas para definir as transformações físicas a fazer nos dados (verificar a secção 4.4 para ter uma ideia do tipo de transformações possíveis de se fazer), validações, *inputs* e *outputs* para bases de dados, ficheiros, etc; os *jobs* (tarefas) usados para orquestrar o fluxo de dados entre transformações e outras tarefas, assim como executar algumas operações úteis do ponto de vista de administração do *data warehouse* (por exemplo, verificar se uma determinada tabela existe na base de dados, executar comandos de SQL, definição de variáveis que são usadas no resto da transformação, etc.). Outra diferença importante entre transformações e tarefas é que os fluxos definidos nas transformações são executados em paralelo enquanto que os fluxos das tarefas são executadas de forma sequencial, daí as tarefas serem indicadas para orientar o fluxo de trabalho enquanto que as transformações são indicadas para processamento de dados.

Em cada um destes artefactos, cada acção é representada por um *step* (passo), cada um com o seu próprio écran de propriedades dependendo do seu tipo. Os passos são depois interligados entre si, ditando o fluxo de dados a ocorrer na transformação. Desde a versão 3.2 do *Pentaho Data Integration* pode-se contar com 101 passos diferentes disponíveis na aplicação, sem contando com *plug-ins* de terceiros.

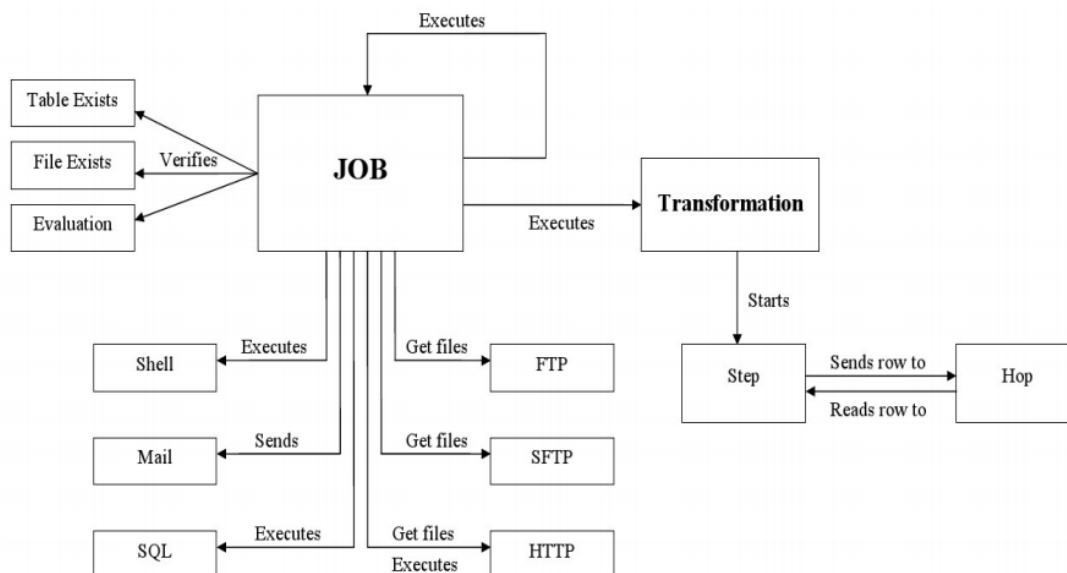


Figura 5.4: Modelo conceptual dos artefactos produzidos pelo Pentaho Data Integration

Para além de ser baseado em modelos, o *Pentaho Data Integration* destaca-se pela sua escalabilidade, dando suporte a criação de *clusters*[63] e de partições (conjuntos de *clusters*) para execução dos processos de ETL repartidos por várias máquinas em rede. Na execução local de transformações, dá também suporte a paralelismo e *multi-threading*, garantindo uma boa performance face a um processamento complexo de dados. A sua arquitectura também é facilmente extensível pelo suporte a implementação de *plug-ins* de terceiros.

O *Pentaho Data Integration* é constituído por um conjunto de várias ferramentas para a modelação, execução e monitorização de um processo de ETL cujo o funcionamento integrado pode-se ver na figura 5.5. Entre elas pode-se encontrar:

- **Spoon** — Um editor gráfico usado para criar *transformations* (transformações) e *jobs* (tarefas). Possui funcionalidades de *debugging*, previsão e teste sobre os artefactos que produz.
- **Carte** — Um servidor que permite a execução remota de transformações e tarefas.
- **Pan** — Uma ferramenta de linha de comando que executa transformações.
- **Kitchen** — Uma ferramenta de linha de comando que executa tarefas.

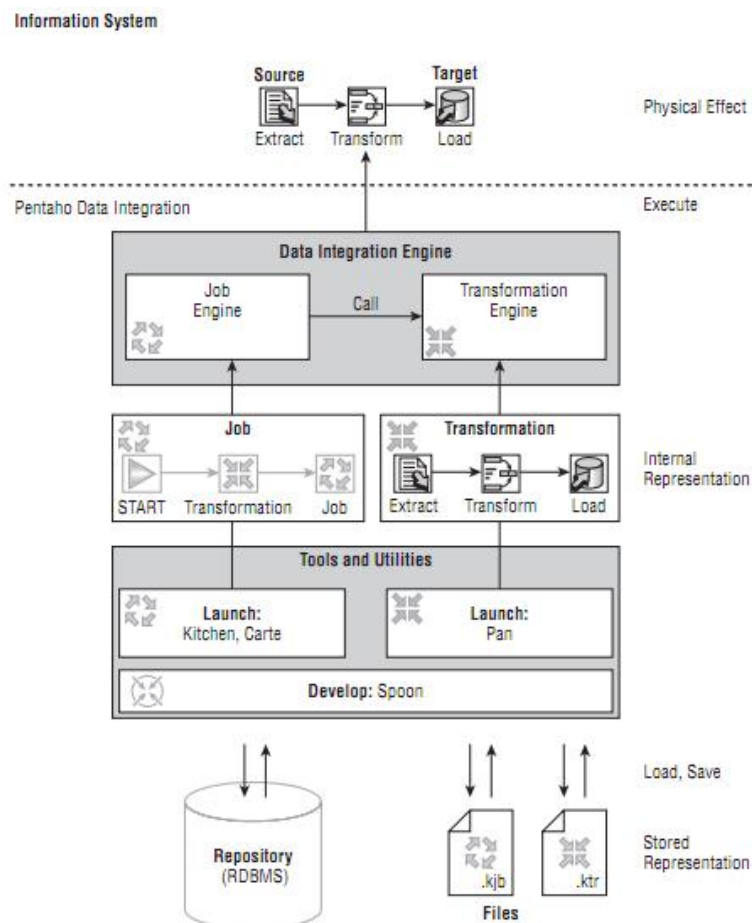


Figura 5.5: Modelo da integração das diferentes ferramentas do Kettle

- **Pentaho Data Integration Console** — Um componente apenas presente na versão comercial do *Pentaho Data Integration*. Permite a definição de alertas, armazenamento de *logs* de execução, controlar a execução remota de tarefas e transformações e obter métricas relacionadas com a performance dos processos de ETL.

Todo o fluxo de trabalho começa pelo editor gráfico *Spoon*, produzindo um conjunto de ficheiros XML de transformações (de extensão *ktr*) e de tarefas (de extensão *kjb*) que representam os vários passos a executar no processo de ETL. Estes ficheiros podem ser guardados no sistema de ficheiros local ou num repositório de ficheiros *Pentaho*. Após obter estes ficheiros, pode-se executá-los localmente através da própria interface do *Spoon* ou através das ferramentas *Pan* ou *Kitchen* para executá-las através da linha de comandos. pode-se também fazer pedidos a um servidor remoto — *Carte* — para executar a transformação através do *Spoon* ou definir esta atribuição de execução entre diversos servidores no próprio modelo criado. Pode-se inclusivamente definir ao nível do modelo o *cluster* ou partição de servidores onde se quer executar uma determinada parte da transformação.

Aspectos Importantes

Na análise efectuada ao *Pentaho Data Integration* e às suas ferramentas, pode-se concluir que esta *suite* está bem preparada para cumprir qualquer requisito de *Business Intelligence* actual, apresentando poucas falhas a nível de funcionalidades e poucas limitações na sua versão grátis *open-source*.

A interface providenciada pelo *Spoon* é simples, podendo ser utilizada por utilizadores orientados ao negócio, mas não deixando de ser bastante parametrizável em aspectos mais técnicos da ferramenta, abrindo portas para requisitos mais complexos de utilização (como é o caso de funcionalidades avançadas de *logging* por exemplo). A modelação dos processos de ETL é um processo relativamente simples, tendo em conta a complexidade típica da especificação de processos de ETL noutras ferramentas semelhantes. A performance é adequada uma vez que o motor de ETL está preparado para execuções em paralelo, assim como execuções distribuídas por diversas máquinas, podendo repartir a carga de processamento mais complexo de processos ETL. Existe bastante documentação sobre as diversas ferramentas disponibilizadas, inclusive uma comunidade sólida, disposta a ajudar em problemas mais concretos.

Relativamente à *Pentaho*, é uma empresa especializada na criação de soluções de *Business Intelligence* e cujo primeiro lançamento de *software* no mercado foi feito em 2007. Trata-se de uma empresa relativamente madura com um seguimento próximo das últimas tendências do mundo de BI (com a adopção da filosofia do *Agile Business Intelligence* e uma aposta forte nas tecnologias *mobile*). A sua presença está relativamente assegurada no mercado de BI, tornando a probabilidade de descontinuação deste *software* bastante baixa.

5.2.2 Talend Open Studio

O *Talend Open Studio*[50] é considerado o maior competidor directo do *Pentaho*, sendo o segundo maior produto *open-source* de ETL presente actualmente no mercado.

Este software, em contraste com o *Pentaho Data integration*, opera internamente como um gerador de código, podendo gerar código *java* e *perl* responsável por instanciar os processos de ETL modelados através da sua interface gráfica.

Esta interface também é diferente, assim como o seu modelo geral de funcionamento, considerando que este estúdio organiza todo o seu arsenal num *workspace* colaborativo, baseando-se na interface do *eclipse*[45]. Dentro desta interface pode-se encontrar um repositório de meta-dados, usado para armazenar definições e configurações para cada processo, e uma área de desenho gráfico *drag-and-drop* de *jobs* (tarefas) orientada a componentes semelhante à alternativa da *Pentaho*. As tarefas depois de criadas são exportadas para um ficheiro comprimido que contém um executável que invoca o código gerado pelo modelo. O elevado número de componentes disponibilizados para modelação de proces-

tos de ETL apresentam uma abordagem mais baixo-nível, exigindo uma especificação mais complexa de cada passo e, conseqüentemente, uma maior curva de aprendizagem por parte do utilizador.

Também apresenta a possibilidade de criar modelos alto-nível de negócio, sendo útil para manter todos os intervenientes no processo a par do que é necessário fazer (embora não haja qualquer interligação com os modelos de ETL desenhados, mantendo um carácter meramente informativo).

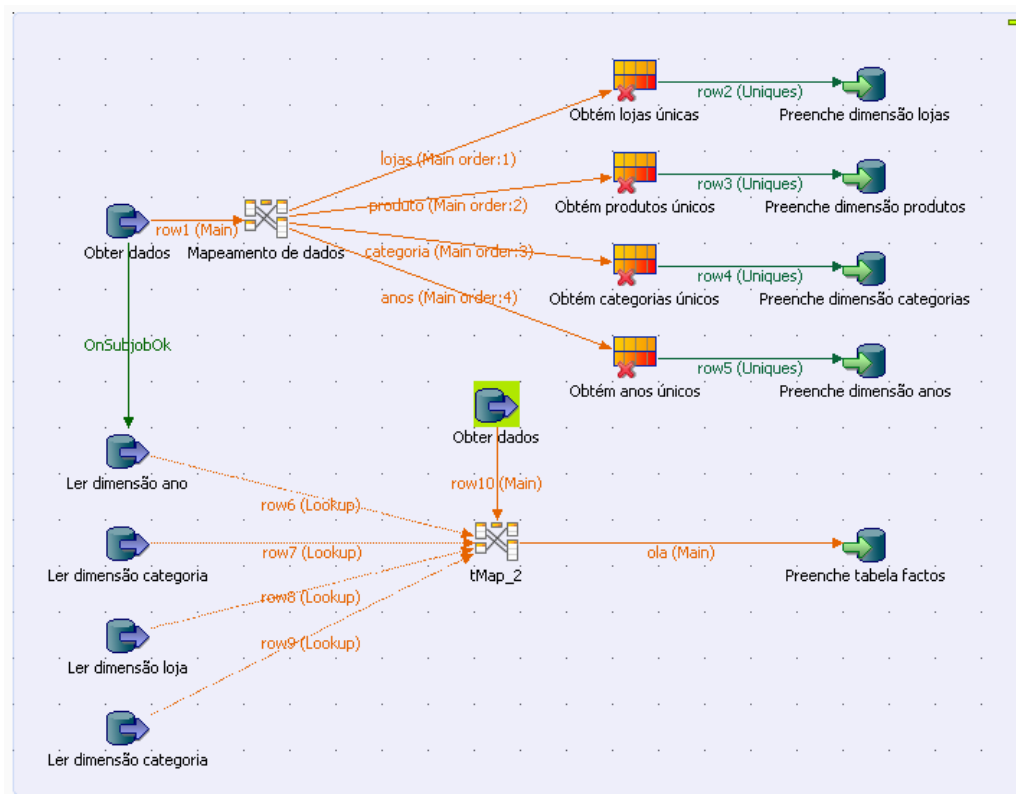


Figura 5.6: Modelo resultante do exemplo ETL pelo Talend Open Studio

Aspectos Importantes

Na análise efectuada ao *Talend Open Studio* foram encontrados alguns *bugs* na especificação do exemplo básico de um processo de ETL. Um desses *bugs* provou ser fatal uma vez que corrompeu o modelo construído até à altura, revelando assim alguma instabilidade da ferramenta.

A curva de aprendizagem foi demasiado acentuada, apesar da presença abundante de documentação e tutoriais básicos de uso da ferramenta. Isto deve-se em parte à abordagem baixo-nível que foi tomada na concepção dos componentes de modelação como foi referido anteriormente. No entanto, apesar destas dificuldades, foi considerado que esta complexidade de especificação poderia vir a ser útil no caso de processos de ETL que

exigem uma granularidade maior. Em termos de performance, pôde-se observar que ficou um pouco atrás comparativamente à alternativa da *Pentaho* na execução do exemplo referido.

O modelo resultante do processo de ETL é consideravelmente mais complexo, exigindo um grande número de passos intermédios para o que foi considerado como um modelo simplista de um *data warehouse*. Em especificações mais complexas é previsível a ilegibilidade do modelo resultante e as dificuldades que daí advém, como por exemplo a manutenção do modelo.

5.2.3 CloverETL

O *CloverETL*[43] é uma ferramenta que foi considerada como uma alternativa viável de execução de processos ETL devido ao seu rápido crescimento nos últimos anos a nível de funcionalidades implementadas e expansão da comunidade por detrás do projecto.

Apresenta algumas características em comum com o *Pentaho Data Integration*, nomeadamente a sua centralização na construção e interpretação de modelos das transformações ETL por um motor. Estes modelos também são similarmente criados através duma interface gráfica, a qual também é baseada no eclipse.

Em termos de funcionalidades, apresenta um conceito que aparenta ser uma tentativa de mistura da facilidade de utilização do *Pentaho Data Integration* na modelação de transformações, em conjunto com a possibilidade de inserção de código customizável nos seus componentes como o *Talend Open Studio*, conseguido através duma linguagem específica do motor do *CloverETL*: o *CloverETL Transformation Language* (CTL2).

O seu modo de funcionamento é inteiramente baseado em meta-dados, quer nos componentes em si, quer nas ligações de dados entre eles, exponenciando ao máximo o factor de reutilização das transformações criadas.

Aspectos Importantes

A versão gratuita do *CloverETL* conseguiu em parte satisfazer os critérios de análise das ferramentas de ETL abordadas nesta investigação.

A sua simplicidade de uso e curva de aprendizagem pouco acentuada ficou praticamente ao mesmo nível do *Pentaho Data Integration*. A documentação do projecto e tutoriais são abundantes e a comunidade online por detrás da ferramenta revelou ser bastante prestável. A *CloverETL Transformation Language* também possibilita uma maior flexibilidade no uso da ferramenta relativamente à alternativa da *Pentaho*.

Durante a avaliação foram encontrados alguns *bugs* de interface que, embora de impacto praticamente nulo, tornava a interface incoerente com o modelo da transformação ETL de exemplo que na altura se estava a criar, mostrando ligações entre componentes gráficos que já tinham sido apagados.

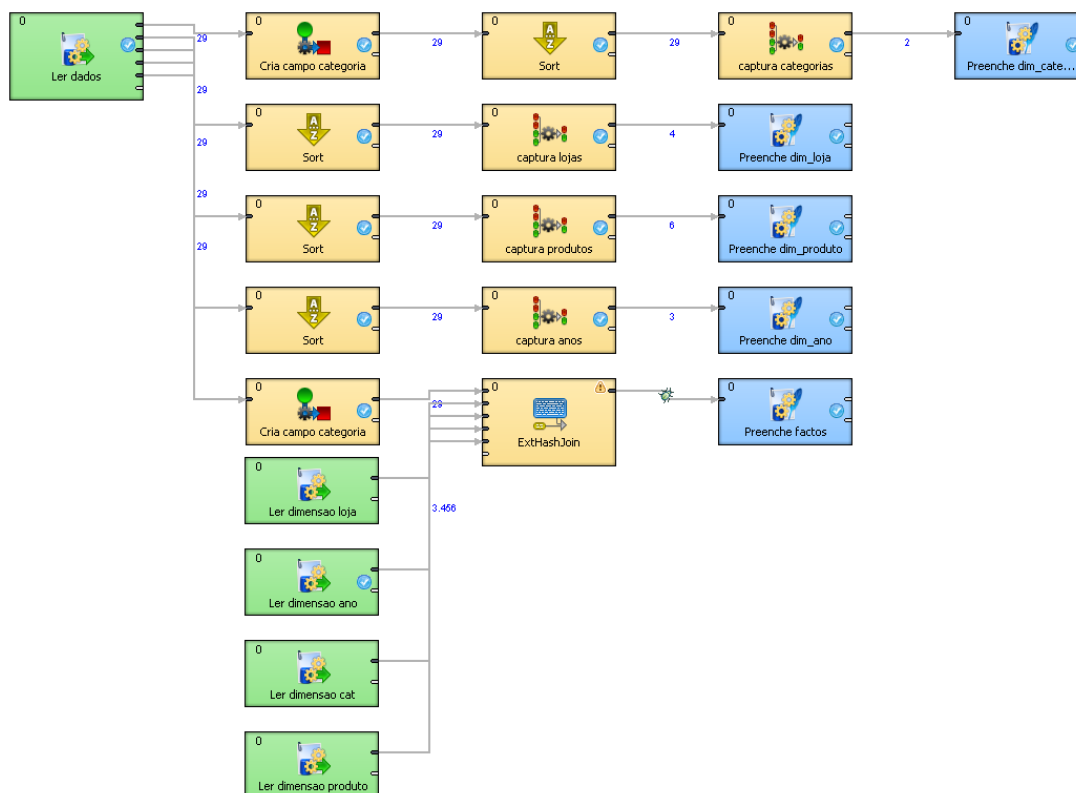


Figura 5.7: Modelo resultante do exemplo ETL pelo CloverETL

Relativamente à empresa por detrás desta ferramenta — *Javlin* — pode-se afirmar que revelou alguma maturidade, tendo atingido a primeira meta do lançamento oficial do *CloverETL* em 2002. Existe também adopção da versão comercial da ferramenta por parte de algumas empresas, o que reduz em parte o risco de descontinuação.

Apesar do *CloverETL* apresentar uma combinação interessante de funcionalidades, a sua versão gratuita ficou um pouco aquém das expectativas, faltando bastantes componentes de design de transformações, assim como outras capacidades interessantes de paralelismo e execução remota presentes nas versões comerciais do produto. Os modelos criados através da versão gratuita também são limitados a vinte componentes, tornando-se inútil na perspectiva do projecto.

5.3 Conclusão da Análise

Após a análise das três ferramentas de ETL abordadas, pode-se concluir que devem adotar o *Pentaho Data Integration* (Kettle) na criação do novo módulo de ETL do SIAG.

A sua simplicidade de uso na modelação de processos de ETL é adequada à utilização por utilizadores orientados ao negócio e não implica uma perda de flexibilidade no seu uso, um mal geral que afecta todas as ferramentas que são demasiado simplificadas nas interfaces e parametrizações possíveis. É uma ferramenta que mesmo na sua versão livre

está pronta para o futuro, com o suporte a execução de processos de ETL distribuída por várias máquinas em rede, aumentando a performance e aliviando eventuais problemas de carga que se iria ter num só servidor centralizado. Para complementar estes aspectos, é uma ferramenta que dispõe de um número elevado de componentes de desenho de processos de ETL, com grande tendência a crescer em cada versão lançada.

A *Pentaho*, empresa que suporta o *Kettle*, tem em vista as últimas tendências do mundo do *Business Performance Management*, com forte investimento no mercado movél e no mercado aliciante do *Agile Business Intelligence* que aparenta ser um bom investimento face aos desafios cada vez mais ambiciosos de rapidez de acesso e disponibilidade à informação dos dias de hoje.

Capítulo 6

Arquitectura e Tecnologias Utilizadas

Neste capítulo irá ser discutida a arquitectura do projecto implementado, assim como o das infraestruturas relevantes adjacentes ao projecto.

6.1 O Sistema Integrado de Apoio à Gestão

6.1.1 A antiga arquitectura

O SIAG possui uma implementação com base no *Apache Struts*[49], obedecendo a um padrão de desenho *Model-View-Controller*[13] representado na figura 6.1. Neste padrão de desenho de software os diferentes componentes são separados em três camadas:

- A camada do **Model** (modelo) — Responsável pelos dados da aplicação e o seu comportamento no domínio da aplicação. É nesta camada que é mantido o estado da aplicação e são alterados os dados consoante os pedidos vindos do controlador.
- A camada da **View** (vista) — Responsável por preparar os dados para interações com os diferentes intervenientes no sistema, através das diferentes interfaces disponibilizadas pela aplicação. É nesta camada que são feitos pedidos ao modelo para consultar o estado da aplicação e são feitos pedidos de interação (mudanças) dos dados ao controlador.
- A camada do **Controller** (controlador) — Responsável por efectuar mudanças de dados sobre o modelo e processar os pedidos feitos pelo utilizador por uma vista específica.

No caso específico do SIAG e na sua antiga arquitectura, a camada das vistas é composta por JSP's (*Java Server Pages*[20]) que contêm *java applets* que desenham os formulários no *browser* dos utilizadores. A camada dos controladores é composta por classes que estendem das acções do *Struts*[49]. Estas acções interligam-se entre as vistas e a camada de negócio do SIAG. É nesta camada que se mantém os diferentes modelos das

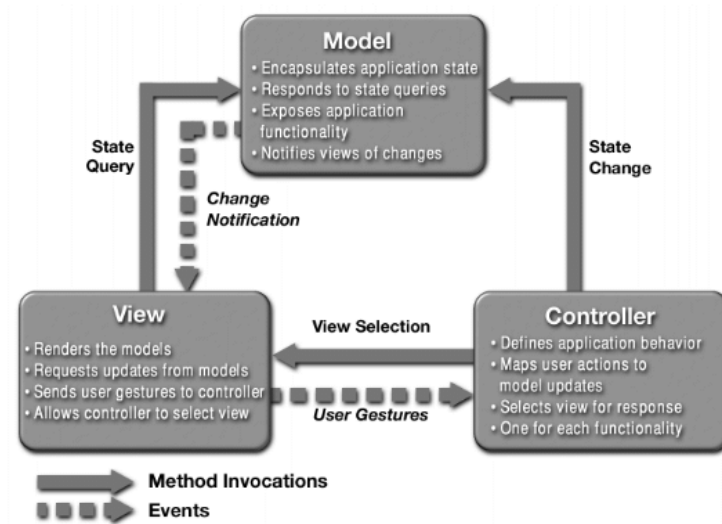


Figura 6.1: Diagrama do padrão Model-View-Controller

várias funcionalidades e objectos, modelos esses que comunicam com a camada de persistência implementada em *Hibernate*[10] através de um *Data Access Object*[67] (DAO). A camada de negócio é composta pelos *Plain Old Java Objects*[84] (pojos) gerados pelo *Hibernate* e por serviços *Spring*[38] que controlam as modificações e gravações dos modelos na base de dados através do DAO. Os pojos servem assim para mapear objectos *java* ao modelo de dados implementado na base de dados, abstraindo da sua implementação física nas diferentes bases de dados suportadas.

6.1.2 A nova arquitectura

Actualmente, a antiga arquitectura encontra-se em expansão através de novos mecanismos que reforçam uma metodologia orientada aos processos de negócio (*Business Driven Development*[59]). Esta nova filosofia de desenvolvimento procura atingir vários objectivos a longo prazo para melhorar o funcionamento do SIAG no contexto empresarial:

- Promover a reutilização do código.
- Promover a agilidade e flexibilidade na implementação de novos módulos.
- Estabeler independência entre os diversos módulos e prevenir problemas futuros de escalabilidade.
- Possibilitar a personalização das funcionalidades da aplicação conforme as necessidades do cliente.

Para lidar com o problema da agilidade de desenvolvimento foi primeiro necessário mudar a forma como eram criadas as interfaces do SIAG. Para tal, foi desenvolvida uma plataforma de *Rapid Application Development*[85] — o *SiagGWT*[3] — que permite a

criação de écrans através de uma interface criada por *drag-n-drop* de componentes de formulário. Para além das validações e comportamentos básicos incluídos nos próprios componentes, é possível expandir o comportamento do écran e dos diversos componentes através de classes *java* designadas por *actions*. Estas *actions* depois de compiladas geram código *javascript* específico para cada *browser*, emulando o comportamento definido no código *java*. O écran pode assim invocar através de eventos despoletados pelos seus componentes métodos definidos nas *actions* de forma a complementar o seu funcionamento normal.

Com a adição desta nova forma de desenvolvimento de interfaces, foi também implementado um padrão de integração empresarial — *Message Broker*[93][11] — sobre JMS[79] (*Java Messaging System*) e *Apache ActiveMQ*[42] para permitir a comunicação síncrona entre os diversos módulos que constituem o SIAG. Neste padrão o cliente e o servidor não precisam de ter conhecimento um do outro, bastando apenas saber o tópico (ou canal) para o qual devem enviar ou escutar novas mensagens. A implementação em si também não necessita de exposição de interfaces adicionais dos serviços que atendem os pedidos JMS. Ambos os lados necessitam de uma única interface comum para escutar e enviar mensagens, simplificando todo o processo de implementação de independência entre clientes e servidores.

Com esta nova forma de comunicação, as funcionalidades do SIAG podem ser divididas em servidores independentes. Cada servidor depende do mesmo projecto base com todas as funções básicas disponíveis no SIAG, reforçando a reutilização de código comum. Para atender as mensagens JMS, são criados um ou mais *workers* que dentro do servidor encaminham os dados anexados à mensagem para o respectivo *engine* (motor) para que possa processar os dados. Criar uma nova funcionalidade torna-se tão fácil quanto acrescentar um novo método ou conjunto de classes no respectivo *engine*. O código fica separado da camada de persistência do SIAG, aumentando a flexibilidade de como os motores podem ser usados em novos paradigmas de negócio. Personalizar uma instalação do SIAG é tão simples como colocar ou retirar um determinado servidor da *release* distribuída aos clientes, visto que a API já vem preparada para esta integração.

Na especificação de novos módulos baseados em comunicação JMS[79], tudo começa na classe abstracta *JmsServletListener*, que actua como um *servlet* de atendimento de mensagens. Esta classe regista um conjunto de classes designadas por *workers* que vão receber as mensagens encaminhadas por um *listener* específico através de um método fixo — *internalProcess()* — que recebe a designação de uma acção e um objecto serializável que pode vir opcionalmente em anexo com a mensagem. O método *internalProcess()* obtém a acção pretendida e desserializa os objectos em anexo, encaminhando o pedido para as respectivas classes de processamento. Iniciar um novo módulo passa então pela implementação de uma classe que extenda o *JmsServletListener* e registe os seus próprios *workers*.

6.2 Módulo SiagBI

O nascimento do *SiagBI* deve-se sobretudo ao alinhamento com as novas metodologias de desenvolvimento do SIAG especificadas na secção 6.1.2. Nesta secção irão ser abordados aspectos arquitecturais importantes que se teve em conta na implementação deste módulo.

6.2.1 Arquitectura

As funcionalidades de BI dependentes das bibliotecas do *Mondrian* e *JPivot* (descritas na secção 6.4.1 e 6.4.1 respectivamente) podem ser divididas em duas categorias: funcionalidades que atendem os pedidos da interface do explorador de cubos OLAP, e funcionalidades da interface de criação de cubos OLAP que geram o código XML que é alimentado ao modelo de dados do *Mondrian* para que os dados de um cubo possam ser explorados.

Para exploração dos dados de um cubo é mantida uma cache sobre os modelos *Mondrian* criados (representantes dos cubos) para os quais foi feito um pedido de exploração. O acesso a estes objectos é feito de forma síncrona para evitar problemas de acesso concorrente ao mesmo cubo OLAP por utilizadores diferentes. Para manter este comportamento e centralizar o acesso a esta cache foi criada uma classe *CacheManager* implementada no padrão de desenho *Singleton*[94] como demonstrado na figura 6.2.

Neste padrão de desenho, o construtor é declarado como privado de forma a restringir a instanciação de novos objectos da classe. O próprio objecto é declarado como uma variável privada dentro da própria classe. A construção do objecto é feita apenas uma vez através do seu constructor privado ou na própria declaração da variável privada que irá apontar para a sua instância. O método *getInstance()* é implementado de forma a devolver sempre a variável interna com a instância da classe. Desta forma, é garantido que existe apenas uma única instância do objecto, perfeito para o cenário de apenas querer manter um único conjunto de estados dos modelos *Mondrian* em memória, sem nunca instanciar um segundo objecto do tipo *CacheManager*.

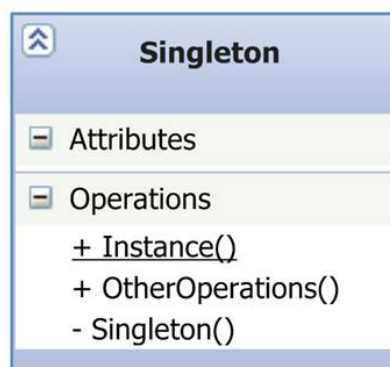


Figura 6.2: Diagrama UML de uma classe que implementa o padrão Singleton

Relativamente às funcionalidades de geração de XML dos cubos *Mondrian*, estas fo-

ram especificadas num motor aparte da classe *MondrianEngine*, a classe *CubeEngine*, motor responsável apenas por saber gerar o código XML usado para definir um cubo OLAP de *Mondrian*. Antes da separação das funcionalidades de BI para o novo módulo, o código XML era em grande parte gerado a partir da camada de persistência de SIAG, uma vez que depende da construção de *queries SQL* e colocá-las no próprio XML. Uma vez que as funcionalidades de BI apenas corriam exclusivamente sobre a camada de persistência do SIAG, estas *queries SQL* eram obtidas a partir de uma extensão (construída no próprio SIAG) da API do *Hibernate* para conseguir obter este tipo de detalhe de mais baixo-nível sobre os *pojos* do SIAG e as transacções efectuadas sobre a base de dados. Para conseguir ultrapassar este problema, foi construído um conjunto de classes para obter informação sobre as tabelas de uma base de dados relacional e as relações entre elas. Este módulo de geração do modelo de dados de uma base de dados relacional foi construído sobre dois padrões: o padrão *Facade*[92] e o padrão *Abstract Factory*[91]. Pode-se analisar o modelo UML parcial do módulo de base de dados criado presente na figura 6.3 (os modelos UML completos estão disponibilizados em anexo).

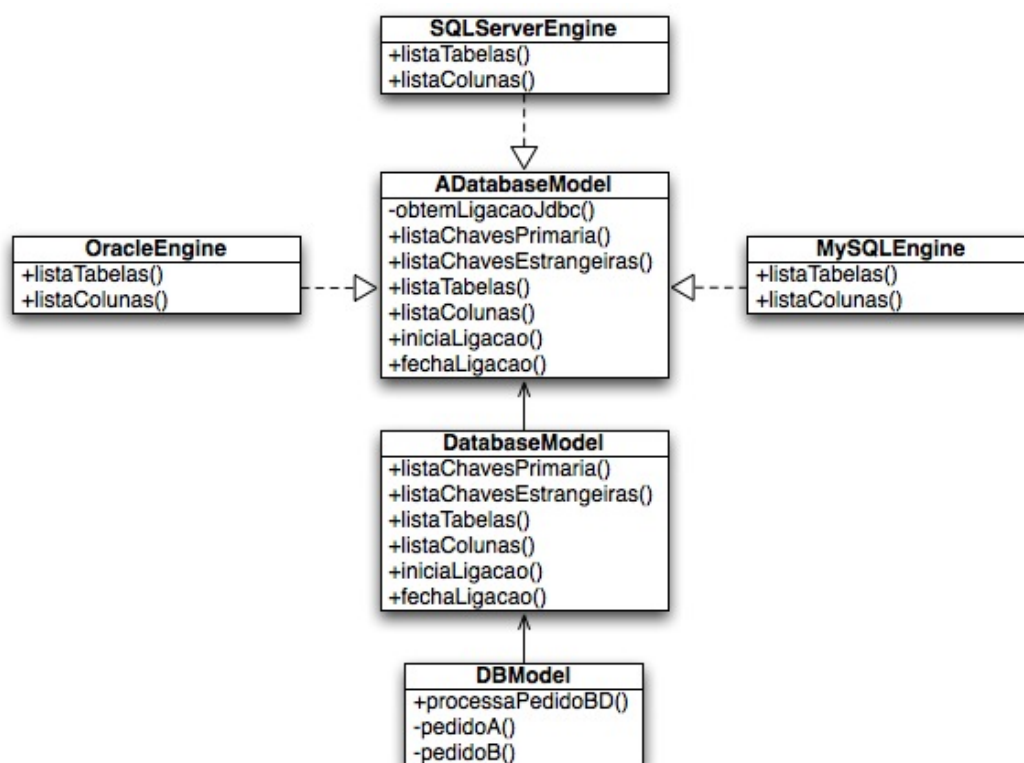


Figura 6.3: Diagrama UML exemplar do módulo de criação de modelos de base de dados do SiagBI

O padrão *Facade*[92] foi implementado a nível da classe *DatabaseEngine* como forma de abstrair por completo a implementação dos métodos específicos de cada tipo de base de dados suportado pelo SIAG. É nesta classe que é feito todo o acesso sobre o modelo

da base de dados em questão e que disponibiliza apenas os métodos necessários à camada de negócio, sem nenhum conhecimento sobre como é feita a implementação das classes abaixo da *DatabaseEngine*. Este conjunto de classes torna-se assim mais fácil de compreender, utilizar e testar na perspectiva de um *software developer*, uma vez que *DatabaseEngine* apenas disponibiliza métodos para tarefas comuns sobre os vários modelos: listagem de tabelas da base de dados, listagem de campos de uma tabela, entre outros. As dependências entre as diversas classes que compõe o módulo são reduzidas, uma vez que o ponto de entrada dos diversos métodos é sempre feito através desta classe, aumentando a flexibilidade do seu uso em outros futuros módulos.

O padrão *Abstract Factory*[91] foi implementado ao nível das classes abaixo de *DatabaseEngine*. Cada tipo de base de dados contém métodos específicos de obter informação através de uma ligação JDBC[78]. Como forma de não complicar a manutenção das diversas classes de suporte aos diversos tipos de base de dados e de não sacrificar flexibilidade futura na implementação feita, cada tipo de base de dados contém o seu próprio motor específico, cada um com a sua implementação diferente. As operações genéricas comuns que são feitas sobre todos os tipos de base de dados (como por exemplo, obter chaves primárias e estrangeiras de uma tabela, o API de abertura e fecho da ligação JDBC, etc.) é disponibilizada pela classe abstracta *ADatabaseEngine*. Cada motor específico de um tipo de base de dados estende desta classe, tendo acesso aos métodos comuns disponibilizados e forçando a reimplementação de métodos específicos do tipo de base de dados em questão. Desta forma, adicionar suporte a um novo tipo de base de dados é tão fácil quanto criar um novo *engine* que estenda de *ADatabaseEngine* e dar conhecimento da nova *engine* à classe *DatabaseEngine* para a saber construir quando for necessária.

6.2.2 Modelo de Dados

O modelo de dados demonstrado na figura 6.4 apenas foi modificado como forma de dar suporte a cubos OLAP cujo modelo de dados é disponibilizado sobre fontes de dados especificadas pelos utilizadores (podendo ser fontes de dados não-SIAG). Para tal, foi criada uma tabela *FonteDados* com toda a informação relativa à conexão JDBC[78] das fontes de dados definidas. Cada fonte de dados também possui um tipo de base de dados específico guardado na tabela *TipoBaseDados*. Esta tabela é responsável por guardar informação relativa a um tipo de base de dados específico como o seu nome específico, uma sigla representante do seu tipo e uma string modelo do URL JDBC usado (para facilitar a sua construção iterativa através das novas interfaces para a criação de novas fontes de dados).

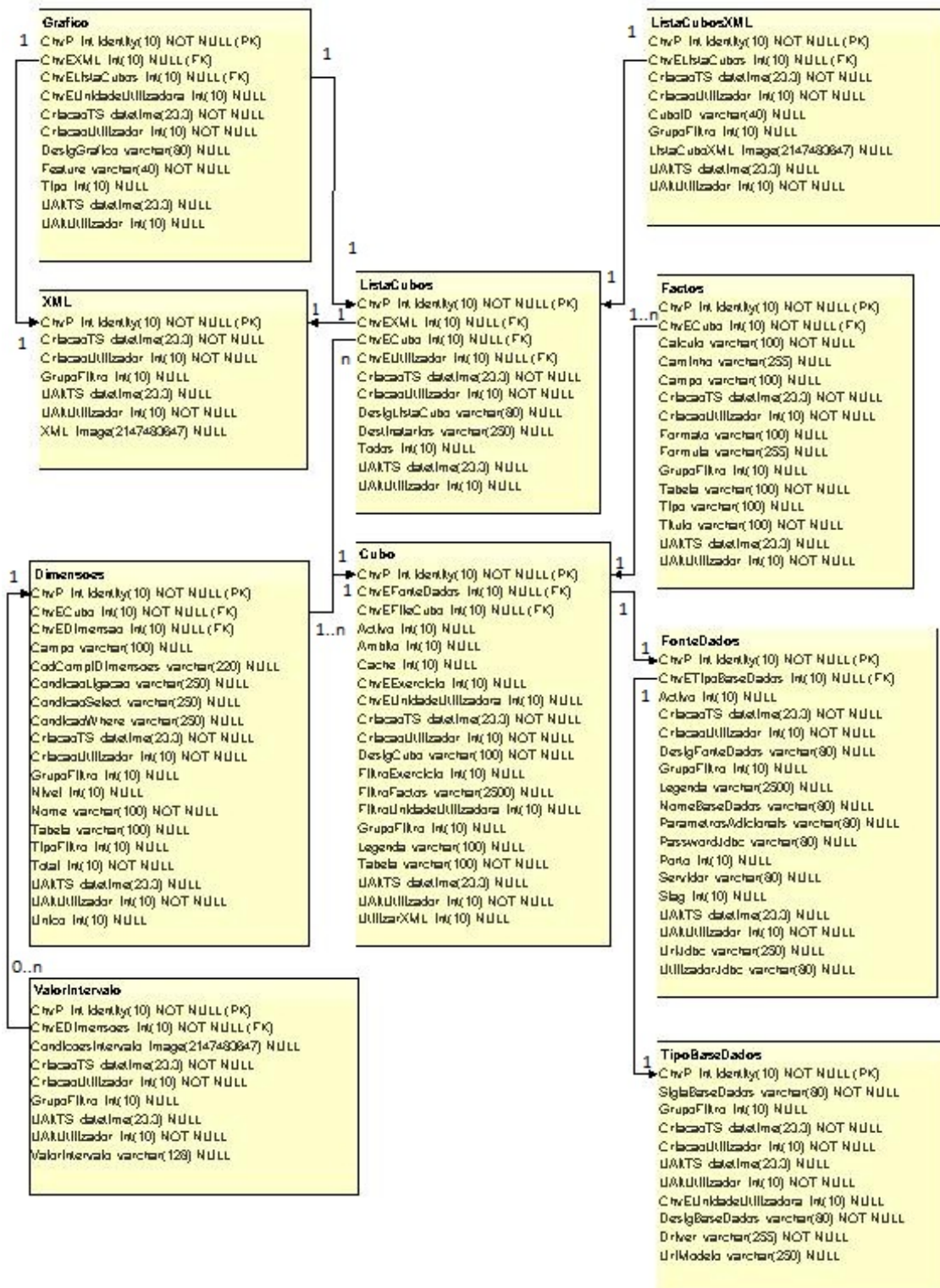


Figura 6.4: Modelo de Dados do módulo SiagBI

6.3 Módulo SiagETL

O nascimento do *SiagETL* deve-se sobretudo ao alinhamento com as novas metodologias de desenvolvimento do SIAG especificadas na secção 6.1.2. Este módulo serve como intermediário entre o SIAG (originados pelas interfaces *GWT* criadas ou por instâncias do processo de *scheduling* de transformações ETL criado sobre a plataforma JBPM[28] existente) e o servidor *Carte* descrito na secção 6.4.1, disponível no pacote do *Pentaho Data Integration*[47].

6.3.1 Arquitectura

Originalmente o módulo *SiagETL* foi surgindo dentro do próprio módulo *SiagBI* na sua forma inicial de integração com o sistema SIAG. No entanto, esta separação de módulos foi necessária pois a manutenção do código produzido iria ser problemática e a escalabilidade futura do SIAG estava em jogo. A separação de ambos os módulos promoveu assim uma melhor organização do código, assim como uma melhor delegação de responsabilidades de cada módulo na perspectiva de negócio.

Dado que nos foi dado o máximo de versatilidade e liberdade de escolha na implementação do novo módulo, tinha à minha disposição toda a pontencialidade do pacote disponibilizado pelo *Pentaho Data Integration*[47] para executar transformações (*transformations*) e tarefas (*jobs*) modeladas através da interface do *Spoon*. Defini então algumas soluções possíveis de integração com o SIAG face a este paradigma:

- Executar transformações e tarefas em *threads* lançadas pelo próprio *SiagETL*, com base no código aberto das ferramentas *Kitchen* e *Pan*. Implementar métodos para obter o estado de execução sobre uma determinada transformação e possibilidade de interromper uma transformação.
- Executar as transformações remotamente a partir do servidor *Carte*. Obter o estado e gerir as transformações através da API disponibilizada pelo servidor.

Dado ambas as opções foi tomada a opção da implementação baseada no servidor *Carte*. Isto deve-se aos seguintes factores:

- O API para gestão de transformações já estava embutido no próprio servidor — Outra implementação exigiria a criação de um API próprio de gestão de threads que executariam as transformações localmente na máquina onde fosse lançado o módulo *SiagETL*. É preferível confiar na experiência dos próprios criadores do *Kettle* no que trata a gestão de transformações, uma vez que uma implementação própria estaria mais sujeita ao aparecimento de *bugs* futuros.
- Redução de dependências e conflitos de bibliotecas java do *Kettle* — A construção de objectos a partir das próprias bibliotecas base do *Kettle* exigiam a presença de

todas as restantes bibliotecas do pacote do *Pentaho Data Integration*, consoante os passos (*steps*) presentes nos ficheiros de transformação gerados pelo *Spoon* (por exemplo, um passo de leitura de dados a partir de uma base de dados *MySQL*[39] exigiria a presença da biblioteca *JDBC*[78] respectiva). Isto em si traria também conflitos com as bibliotecas do *Mondrian* já usadas no módulo *SiagBI* caso fossem lançados ambos os módulos na mesma instância de *JBossjboss*, uma vez que o *Kettle* possui passos para leitura de dados de um cubo *Mondrian*.

- Redução de complexidade da implementação — A implementação baseada no *Carte* apenas exigiria a presença das bibliotecas base do *Kettle*, assim como as bibliotecas disponibilizadas para realizar pedidos HTTP ao servidor, reduzindo assim a complexidade geral da implementação do novo módulo.
- Melhor escalabilidade — Ao implementar o novo módulo sobre o *Carte*, está-se a preparar o módulo de ETL para futuras expansões oferecendo maior capacidade de processamento e maior número de máquinas para atender os pedidos de processos ETL. A arquitectura do *Kettle* está preparada para execução distribuída de transformações ETL com o uso do *Carte*. Na implementação do novo módulo, teve-se em conta uma única instância do servidor *Carte*. No entanto, nada impede no futuro de estender esta arquitectura para incorporar mais servidores para distribuir o a carga de processamento de processos ETL.

Relativamente à camada de persistência e integração com o SIAG, tudo é feito à semelhança do módulo *SiagBI*. A camada de persistência apenas se encontra do lado do SIAG. Toda a comunicação JMS entre o *SiagETL* e o SIAG é feita através da criação de objectos representantes do *pojos* gerados pelo *Hibernate*.

6.3.2 Modelo de Dados

O modelo de dados do novo módulo de ETL demonstrado na figura 6.5 foi criado de raiz, sem qualquer ligação ao modelo de dados já existente do SIAG. Tudo começa na criação de uma transformação guardada na tabela *TransformacaoETL*. Cada transformação ETL no sistema SIAG possui uma designação especificada pelo utilizador, assim como um nome único gerado pelo SIAG com base nesta designação (necessário devido à falta de tratamento de *encoding* de caracteres não UTF-8 por parte do *Carte*). Cada transformação ETL tem a ligação com vários ficheiros XML gerados pelo *Spoon* que são guardados na tabela *TransformacaoETLFicheiro*. No conjunto de ficheiros da transformação ETL, existe um marcado como o ficheiro inicial, ficheiro esse que dá início a todo o fluxo de transformações ETL e que interliga todos os ficheiros relativos a uma transformação. Uma transformação também tem a ligação com várias variáveis que são lidas a partir dos ficheiros guardados e que são guardadas na tabela *TransformacaoETLVariavel*. A interface criada permite a leitura da informação contida em *steps* do tipo *Set Variables* e a

redefinição dinâmica dos valores encontrados nesses *steps* a partir da informação guardada nesta tabela. Sempre que uma transformação é enviada para o *Carte*, os ficheiros são lidos e reescritos com as variáveis definidas pelo utilizador no sistema SIAG, permitindo uma melhor parametrização das transformações e evitando a substituição constante de ficheiros do lado do SIAG.

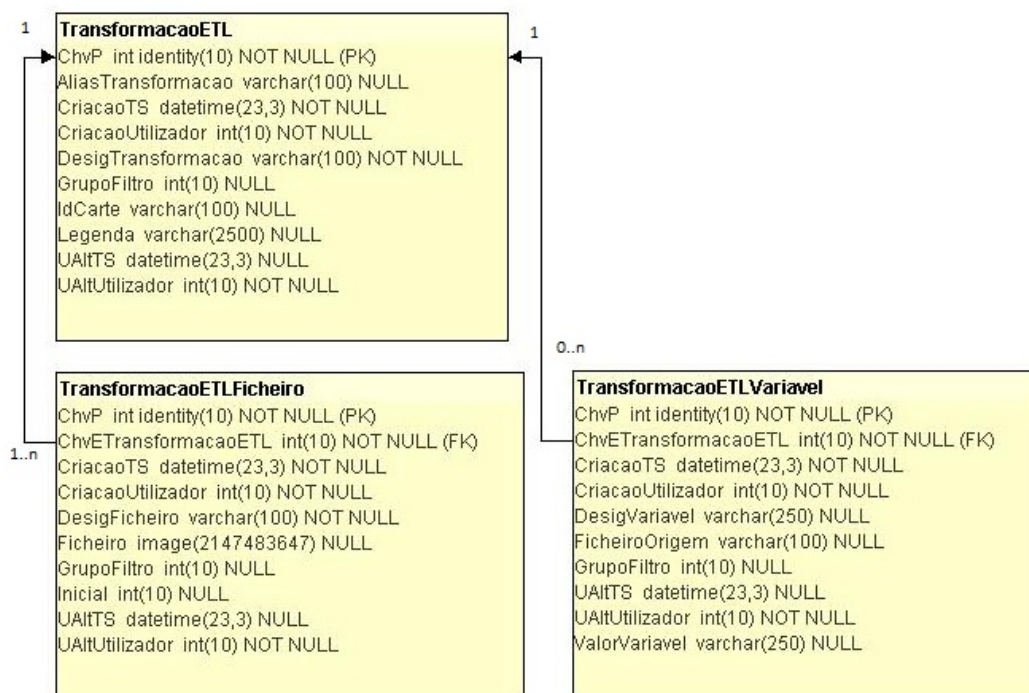


Figura 6.5: Modelo de Dados do módulo SiagETL

6.4 Tecnologias Utilizadas

6.4.1 Java Messaging System

O *Java Messaging System* é um sistema de mensagens, criado com o propósito de permitir a criação de aplicações Java flexíveis e escaláveis. É usado pelo sistema SIAG para a comunicação desacoplada com os seus diversos módulos construídos sobre a nova arquitectura especificada na secção 6.1.2. A grande vantagem do uso desta tecnologia passa pelo uso de uma única interface comum entre cliente e servidor, não necessitando de conhecimento específico da implementação de um determinado módulo, nem a exposição de interfaces adicionais no novo módulo para que se estabeleça comunicação entre ambos os lados. A comunicação é baseada em tópicos, bastando a um cliente ter conhecimento do tópico a que um determinado módulo servidor está à escuta para que consiga comunicar com este módulo.

Mondrian

O *Mondrian* é um servidor OLAP[83] *open-source* baseado em Java criado por *Julian Hyde*. Este servidor suporta *queries MDX* (*MultiDimensional eXpressions*[81]), XMLA (*XML for Analysis*[55]) e especificações *Olap4j*, uma API equivalente à providenciada pelos *drivers JDBC* para o suporte específico a estrutura de dados multidimensionais. No caso específico de SIAG, o *Mondrian* é usado para processar as *queries MDX* construídas dinamicamente pela interface do explorador de cubos do módulo de BI.

JPivot

O *JPivot* é um conjunto de bibliotecas JSP[20] capaz de realizar operações comuns de explorações de dados OLAP tais como *drill-through*, *drill-down*, *slice-and-dice*, *roll-up*, entre outros. É usado de forma integrada com o servidor *Mondrian* para efectuar explorações de dados sobre cubos OLAP e também providencia suporte a fontes de dados XMLA (*XML for Analysis*[55]).

Carte

O *Carte* é um servidor implementado sobre *Jetty*[29] incluído no pacote do *Pentaho Data Integration*[47] descrito na secção 5.2.1. Permite a execução e gestão remota de transformações ETL modeladas pela interface gráfica do *Spoon* a partir de pedidos HTTP. Este servidor também está preparado para ser integrado em *clusters* e partições de servidores ETL de forma escalável.

Fitnessse

O *Fitnessse* é uma plataforma de testes funcionais cuja organização é baseada no formato *wiki* com suporte a plugins denominados de *fixtures*. É usada no contexto SIAG (integrada no projecto *SiagFitnessse*) como forma de forçar a execução de "baterias" de testes funcionais a cada *release* do SIAG de forma a detectar *bugs* antes da nova *release* ser disponibilizada aos clientes. É usada em conjunto com uma *fixture* com suporte a chamada de serviços *Hibernate* — a *Service Fixture* — de forma a poder testar a camada de negócio do SIAG no que toca a criação e alteração de *pojos* na base de dados do SIAG.

6.5 Testes

Nesta secção irão ser abordados duma perspectiva alto-nível os testes realizados sobre ambos os módulos criados.

6.5.1 Testes sobre o módulo SiagBI

Após a implementação do módulo *SiagBI* foi necessário definir alguns testes funcionais de forma a garantir que a geração de XML dos cubos OLAP sobre as nova especificação de fontes de dados era feita correctamente. Para tal foram criados três testes funcionais sobre a plataforma *fitnessse*, cada um com o seu propósito específico:

- *GeraCuboExecucaoDespesa* — Gera um cubo com dimensões de classificação de várias tabelas no sistema SIAG para análise de execução de despesa. Este tipo de dimensões apresentam uma estrutura recursiva a nível dos próprios registos da tabela *Estrutura* onde é feita esta classificação.
- *GeraCuboMantis* — Gera um cubo sobre a própria base de dados *MySQL* do *Mantis*[17] relativa ao tratamento de correcções e pedidos sobre o SIAG-AP. Este cubo é usado para análise de casos do SIAG.
- *GeraCuboVendas* — Gera um cubo típico com dimensões para análise de vendas.

6.5.2 Testes sobre o módulo SiagETL

Após a implementação do módulo *SiagETL* foi necessário definir alguns testes funcionais de forma a garantir que a API criada sobre o servidor *Carte* estava a funcionar correctamente, assim como a criação dos *pojos* de *Hibernate* relativos a uma transformação ETL. Para tal, foram criados três testes funcionais sobre a plataforma *fitnessse*, cada um com o seu propósito específico:

- *ExecutaTransformacao* — Testa o método de execução síncrona de uma transformação ETL usado no processo de *scheduling* sobre a plataforma existente de JBPM[28]. Este método usufrui da restante API criada para comunicação com o *Carte*, testando assim num só método o envio da transformação para o servidor, pedido de execução da transformação, obter estado da transformação e remoção da transformação no .
- *GeraAlias* — Testa a geração de um alias único na gravação de uma transformação, sem caracteres estranhos que possam induzir o *Carte* em erro.
- *TestaFicheiroInicial* — Testa a validação feita sobre a gravação da transformação que impede que uma transformação seja guardada sem um único ficheiro inicial declarado.

Capítulo 7

Trabalho Realizado

Neste capítulo irá ser exposto em mais detalhe todo o trabalho realizado na fase de implementação do projecto, abordando em primeiro lugar o módulo do *SiagBI* e terminando com a implementação do módulo *SiagETL*.

7.1 Criação do módulo *SiagBI*

Inicialmente foi criado um servidor apenas com o código de base SIAG para atendimento de mensagens JMS[79] e foi iniciada a passagem todas as classes e código dependentes de bibliotecas *Mondrian* e *JPivot* para dentro do domínio do *SiagBI*.

No local onde deveria estar o código específico de BI são criados objectos simples com os dados necessários para efectuar essa parte do processamento e é feito um pedido síncrono de JMS ao servidor de BI para processar a parte do algoritmo correspondente.

Cada uma destas chamadas JMS ao servidor começou inicialmente por corresponder a um método do lado do *SiagBI*. Eventualmente, após terem sido retiradas todas as dependências das bibliotecas de *Mondrian*[31] e *JPivot*[30], foi feita uma análise aos diferentes algoritmos da camada de negócio e o código foi optimizado para reduzir o número de pedidos JMS ao servidor e a complexidade do algoritmo em si.

7.1.1 Organização dos *packages* Java

Os *packages* foram organizados da seguinte forma:

- **pt.gedi.bi.cache** — Pacote com classes relativas à cache do servidor. Neste caso não foram necessárias classes adicionais à excepção da classe principal, *CacheManager*, classe gestora da cache dos modelos gerados pelas bibliotecas do *Mondrian* de forma a optimizar a performance de pedidos de exploração de dados de um cubo.
- **pt.gedi.bi.database** — Pacote com as classes relativas à geração do modelo de uma base de dados relacional. É este novo sub-módulo que permite a independência da camada de persistência do SIAG baseada em *Hibernate*[10].

- **pt.gedi.bi.jms.listeners** — Pacote com os *listeners* que registam *workers* para atender pedidos de JMS. O *SiagBI* apenas possui uma classe — *BIJmsServletListener* — que regista um único *worker* — *BiWorker*.
- **pt.gedi.bi.mondrian** — Pacote com as classes responsáveis por pedidos de exploração de dados de um cubo OLAP.
- **pt.gedi.bi.mondrian.xml** — Pacote com as classes responsáveis por pedidos de geração de um ficheiro XML representante de um schema mondrian para um cubo OLAP.
- **pt.gedi.bi.utils** — Classes utilitárias que actuam sobre as restantes classes do módulo de BI e que não são genéricas o suficiente para poderem ser colocadas no código base dos novos módulos JMS do SIAG.
- **pt.gedi.bi.workers** — Pacote com os *workers* que atendem pedidos de JMS. No caso do *SiagBI* apenas contém um *worker* — o *BiWorker*.

7.1.2 Passagem das funcionalidades de BI para o novo módulo

Como primeira tarefa do projecto teria-se de refactorizar todo o código relativo a funcionalidades de BI dependente das bibliotecas *JPivot* e *Mondrian* para o novo módulo. No entanto, durante este processo era necessário assegurar que não existissem dependências com a camada de persistência do SIAG e que ao mesmo tempo a arquitectura existente na construção de cubos OLAP pudesse ser extendida para possibilitar a construção de cubos sobre fontes de dados externas que não fossem do SIAG.

Para tal, as responsabilidades do novo módulo foram divididas em dois sub-conjuntos de funcionalidades distintas: As funcionalidades que atendiam os pedidos da interface do explorador de cubos OLAP (pedidos de dados a um cubo) e as funcionalidades da interface de criação de cubos OLAP que geravam o código XML que é "alimentado" às bibliotecas do *Mondrian*. Os pedidos de dados a um cubo passaram a ser processados pela classe *MondrianEngine*, enquanto que a geração de XML de *schemas Mondrian* passaram a ser processado pela classe *CubeEngine*. Cada tipo de pedido específico foi atribuído uma constante comum entre o *SiagBI* e o SIAG para facilitar o mapeamento de funcionalidades entre ambos os lados e melhorar a manutenção do código.

Tudo começa no arranque do servidor, onde a classe *BiWorker* é registada pela classe *BIJmsServletListener* como classe que atende pedidos JMS com um determinado tópico. Cabe depois ao *BiWorker* encaminhar o conteúdo das mensagens JMS para os respectivos *engines* para serem processadas.

7.1.3 Independência de Fontes de Dados

O segundo passo na criação do novo módulo foi o estabelecimento de independência de fontes de dados nos cubos de *Mondrian*. Podia-se encontrar esta dependência em dois pontos no código: No carregamento de um *schema Mondrian* (especificava-se o url JDBC sobre o qual um cubo actuava) e na geração do próprio XML do *schema Mondrian*.

No modo de utilização clássico do *Mondrian*, num contexto de exploração de dados de um *data warehouse*, bastaria definir as colunas e tabelas de uma base de dados directamente para que se conseguisse definir a tabela de factos, as colunas de ligação com as diversas dimensões especificadas e as colunas que servem de medidas na tabela de factos, sem qualquer tipo de especificação complexa em *queries SQL* uma vez que o modelo de dados do *data warehouse* já obedecia a um esquema em estrela ou em floco de neve.

No contexto do SIAG, o modelo de dados em esquema em estrela é criado artificialmente através de *queries SQL* sobre o modelo de dados SIAG especificadas no próprio XML do *schema Mondrian*. Um cubo OLAP no contexto do SIAG é criado primeiro pela especificação de uma tabela de factos e um conjunto de colunas desta tabela da base de dados SIAG que servem de medidas do cubo. As dimensões são depois especificadas ao escolher um caminho entre os vários caminhos possíveis entre a tabela de factos escolhida e uma determinada tabela alvo que será a nossa dimensão. As *queries SQL* que compõem o cubo eram obtidas através de uma extensão do próprio API do *Hibernate* que permitia obter detalhes transaccionais da base de dados onde actua, gerando a tabela de factos de um cubo através de um conjunto complexo de *joins*[53] entre a tabela central de factos e o conjunto de todas as tabelas especificadas no caminho de uma dimensão.

Foi então necessário a criação de um novo sub-módulo para gerar modelos genéricos de bases de dados relacionais que fosse capaz de obter informação de relações entre tabelas e informação de campos de uma tabela. A geração de XML iria então passar a actuar sobre estes modelos para saber gerar as mesmas *queries SQL* necessárias ao *schema mondrian* para obter dados de um cubo.

Para além desta geração de modelos de base de dados, foi também necessário adicionar suporte na camada de persistência do SIAG para a definição de fontes de dados como indica o modelo de dados na secção 6.2.2. No primeiro arranque do SIAG-AP na nova versão (com a nova forma de interacção com o módulo de BI) é adicionada uma fonte de dados por omissão correspondente à base de dados sobre a qual executa o SIAG. Para além desta fonte de dados por omissão, é dada a possibilidade de adicionar outras fontes de dados através da nova interface demonstrada nas secções seguintes deste documento.

Criação do módulo *database*

O módulo *database* foi criado, tal como foi dito anteriormente, para se conseguir mapear o modelo de uma base de dados relacional (referenciada neste caso por um url JDBC) de modo a conseguir obter informação sobre como as tabelas estão associadas entre si (ob-

tendo as relações *one-to-many* e *many-to-one* entre tabelas) e que campos e chaves contêm uma determinada tabela. Este modelo iria então servir para obter caminhos possíveis entre tabelas para popular componentes da interface de criação de cubos, assim como obter chaves primárias e estrangeiras necessárias para conseguir gerar os *joins* de SQL que compõe o XML de um cubo. O módulo também teria de suportar os mesmos tipos de bases de dados suportados até à data pelo SIAG — *Microsoft SQL Server*[41], *MySQL*[39] e *Oracle*[40] — ser flexível o suficiente para conseguir suportar outros tipos de base de dados futuramente sem grande esforço por parte dos *developers* e poder vir a ser integrada noutros contextos de utilização, não podendo estar dependente do *SiagBI*. A arquitectura adoptada para resolver estas questões é discutida na secção 6.2.1.

Para tal, cabia na nossa decisão optar entre duas implementações possíveis: obter o modelo da base de dados através de diversas *queries* SQL genéricas *hard-coded* no código produzido, específicas a cada tipo de base de dados suportado ou obter o modelo através da API genérica JDBC[5] e os meta-dados devolvidos. Inicialmente foi escolhida a última implementação baseada nos meta-dados JDBC, embora se viesse a descobrir através da própria tarefa de implementação do módulo que a API JDBC, apesar de ser genérica nas suas chamadas, a devolução de resultados era específica ao tipo de base de dados em questão, anulando as vantagens de ter um único componente genérico de base de dados para todos os tipos possíveis. Em adição a este problema, as chamadas de algumas funções da API eram demasiado lentas, como era o caso das bases de dados *Oracle* na obtenção de chaves estrangeiras, entre outros casos específicos.

Face a estes problemas, acabou por se adoptar uma implementação híbrida. Algumas das funções cuja performance fosse melhor iriam ser obtidas genericamente através dos meta-dados JDBC, enquanto que as chamadas cuja performance estivesse degradada segundo os nossos testes iriam ser implementadas sobre *queries* SQL optimizadas para o efeito. Este passo tornou possível obter um modelo de base de dados de SIAG (composto por quase mil tabelas) num tempo máximo de cerca de sete segundos, independentemente do tipo de base de dados em questão.

7.1.4 Actualização das Bibliotecas *Mondrian* e *JPivot*

Como fase final de implementação do novo módulo, as bibliotecas de *Mondrian*[31] e *JPivot*[30] foram actualizadas para a última versão estável disponibilizada online — a versão 3.2.1.13885. Para tal, as bibliotecas antigas foram substituídas e foram feitos alguns testes funcionais e correcções sobre o módulo de BI para garantir que o bom funcionamento das bibliotecas face a nossa implementação específica.

7.1.5 Criação da interface GWT

Em paralelo com a passagem de código para o *SiagBI* foi dado início à criação da nova interface GWT desenvolvida sobre a nova plataforma de construção de ecrãs *SiagGWT*. Esta tarefa começou pela definição da interface de criação de fontes de dados mostrada na figura 7.1. Neste écran foi criado um sistema de criação iterativa do URL JDBC[79] necessário para obter informação de uma base de dados através do módulo *database* anteriormente descrito neste capítulo. À medida que o utilizador preenche os campos do formulário com as credenciais de acesso à base de dados, o URL é construído visivelmente num campo não-editável com base num "URL modelo" guardado para cada tipo de base de dados suportada pelo SIAG. Para além da construção iterativa do URL, também foi implementado um botão de teste de ligação à base de dados que envia pedidos JMS[79] ao *SiagBI* para validá-la. O preenchimento dos campos e outras verificações são depois validados ao nível do código *javascript* compilado que o ecrã invoca e são depois validados novamente quando o respectivo *pojo* do tipo *FonteDados* é criado na camada de negócio do SIAG, antes de ser efectivamente guardado na base de dados.

The screenshot shows a web browser window titled "SIAG - Nova Fonte de Dados". The address bar shows the URL "http://localhost:8080/siagGWT?gui_interface=screen&xmlFilePath=fsfontedados&use". The form contains the following fields and values:

- Designação: Fonte de Dados Siag
- Legenda: Esta é a fonte de dados criada por omissão para cubos que actuam sobre a base de dados SIAG. Por favor preencha a respectiva password da base de dados e teste a ligação antes de guardar as novas definições.
- Activa: ☒
- SIAG: ☒
- Tipo de base de dados: SQLServer
- Nome de utilizador: siag
- Código de acesso:
- Servidor: winpreprod.gedi.pt
- Porto: 1433
- Nome de base de dados: siagin
- Parâmetros adicionais: ;tds=8.0;lastupdatecount=true
- Url JDBC: jdbc:tds:sqlserver://winpreprod.gedi.pt:1433/siagin;user=siag;password=....;tds=8.0;lastupdatecount=true

At the bottom of the form are two buttons: "Testar Ligação" and "Limpar Cache".

Figura 7.1: Novo ecrã de criação de fontes de dados

Para a implementação do ecrã de criação de cubos foi necessário implementar uma

nova forma de escolher a fonte de dados associada a um determinado cubo. Para o efeito, é utilizado o que é designado no âmbito SIAG por *auxiliares*. Estes auxiliares são janelas *pop-up* que permitem ao utilizador escolher um ou mais registos já guardados na base de dados de uma determinada tabela escolhida na própria definição do ecrã. Foi também definido um ecrã genérico *FSFormula* que é reutilizado nos vários separadores de ecrã onde é necessário especificar tabelas ou caminhos entre tabelas (como é o caso da especificação das dimensões da figura 7.2). O ecrã vai populando os seus componentes através de pedidos JMS ao *SiagBI* conforme necessário sobre os modelos de base de dados gerados pelo sub-módulo *database*, obtendo assim caminhos possíveis entre tabelas, tabelas associadas (por relações *one-to-many* ou *many-to-one*) com uma determinada tabela, entre outras informações necessárias na interface.

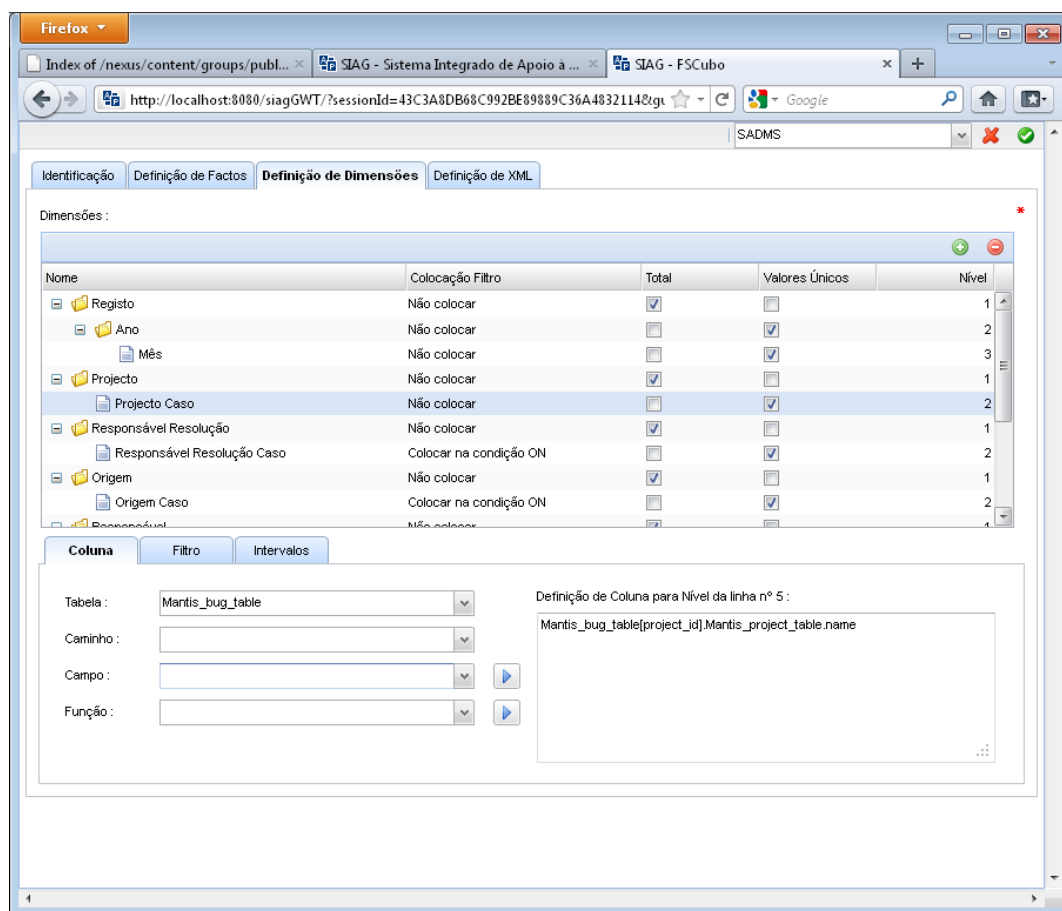


Figura 7.2: Novo ecrã de criação de cubos

7.2 Criação do módulo SiagETL

Após alguns testes manuais sobre as próprias bibliotecas do *Pentaho Data Integration*, começou-se inicialmente por implementar, à semelhança da arquitectura dos novos módulos de SIAG descritos na secção 6.1.2, o esqueleto da arquitectura baseada em JMS, criando

as classes que registam o *worker* de atendimento de mensagens JMS no início do servidor — *ETLWorker*. Dado que em termos de funcionalidades apenas se teria de estabelecer um elo de ligação entre o sistema SIAG e o servidor *Carte* para gestão e execução de transformações ETL, foi apenas necessário a criação de um motor de processamento — o *KettleEngine* — e algumas classes de suporte no processamento dos ficheiros XML produzidos pelo *Spoon*.

7.2.1 Organização dos *packages* Java

Os *packages* foram organizados da seguinte forma:

- **pt.gedi.etl.cache** — Pacote com classes relativas à cache do servidor. Neste caso não foram necessárias classes adicionais à excepção da classe principal, *CacheManager*, classe gestora da cache de ficheiros que são enviados para o SIAG através da interface. Esta cache é necessária uma vez que o utilizador pode fazer pedidos de extracção de variáveis de ficheiros que ainda não estão guardados na base de dados, acabados de enviar da interface.
- **pt.gedi.etl.jms.listeners** — Pacote com os *listeners* que registam *workers* para atender pedidos de JMS. O *SiagETL* apenas possui uma classe — *ETLJmsServletListener* — que regista um único *worker* — *ETLWorker*.
- **pt.gedi.etl.kettle** — Pacote com as classes responsáveis por interacção com o *Carte*, assim como pedidos vindos da interface de extracção de variáveis.
- **pt.gedi.etl.utils** — Classes utilitárias que actuam sobre as restantes classes do módulo de ETL e que não são genéricas o suficiente para poderem ser colocadas no código base dos novos módulos JMS do SIAG.
- **pt.gedi.etl.workers** — Pacote com os *workers* que atendem pedidos de JMS. No caso do *SiagETL* apenas contém um *worker* — o *ETLWorker*.

7.2.2 Integração do *Pentaho Data Integration*

A integração com o *Pentaho Data Integration* foi implementada com base na comunicação por pedidos HTTP com o servidor *Carte* presente no pacote do *Pentaho Data Integration* conforme discutido na secção 6.3.1. Para tal, foi implementado um API que permitisse:

- Extracção de variáveis de ficheiros XML de transformações criados pelo *Spoon*.
- Substituição dinâmica de variáveis nos ficheiros XML ao serem enviados para o *Carte*, conforme a especificação do utilizador na interface.
- Envio de uma transformação (e o seu conjunto de ficheiros) para o servidor *Carte*.

- Pedido de remoção de uma transformação no *Carte*.
- Pedido assíncrono de execução de uma transformação no *Carte*.
- Pedido síncrono de execução de uma transformação no *Carte* (usado em pedidos originados pelo processo JBPM de *scheduling* de transformações, para que um utilizador seja notificado de erros caso algo corra mal na execução).
- Pedido do estado geral de execução das transformações no servidor.
- Pedido do estado detalhado de execução de uma determinada transformação para efeitos de *debugging*.

Esta API iria complementar os serviços de *Spring*[38] implementados acima da camada de persistência com o modelo de dados averiguado na secção 6.3.2.

Extracção e Substituição de Variáveis nos ficheiros de uma Transformação

A extracção e a substituição de variáveis foi implementada através do uso de expressões de *XPath*[54] parametrizadas numa classe de constantes do *Pentaho Data Integration*. Desta forma, a análise sintática dos ficheiros XML das transformações é genérica o suficiente para conseguir abordar facilmente novas futuras implementações sintáticas dos ficheiros, sem que esteja *hard-coded* no código. Estas variáveis são lidas pela interface a pedido do utilizador, procurando passos do tipo *Set Variables* nos ficheiros da transformação e extraindo a informação da instanciação da variável. O utilizador pode então redefinir o valor de uma variável presente nos ficheiros, guardando esta informação do lado da camada de persistência do SIAG. Uma transformação, ao ser enviada para o *Carte*, irá ser automaticamente processada para substituir a variável pelo valor que o utilizador especificou na interface, tornando assim as transformações mais parametrizáveis.

Estes pedidos de extracção de variáveis também foram melhorados para poderem ser feitos sobre uma transformação ETL que ainda não foi gravada no SIAG. Para tal, um ficheiro XML assim que é enviado através da interface GWT é mantido em cache no sistema de ficheiros temporário do *SiagETL*. A cache é mantida por um mecanismo baseado em *timestamps* para evitar a acumulação de ficheiros temporários, cache essa também parametrizável através do ficheiro de constantes do *Pentaho Data Integration*.

Gestão de Transformações no *Carte*

Na implementação dos restantes métodos de gestão de transformações no *Carte* foi necessário inspecionar o código fonte do *Pentaho Data Integration*[47] disponível na web, uma vez que as primeiras experiências feitas sobre o servidor revelaram algumas inconsistências na própria API do *Carte* (por exemplo, envio de uma tarefa (*job*) implica a

sua execução também seguida do envio, enquanto que uma transformação (*transformation*) apenas envia e não executa), assim como alguns *bugs* menores e limitações no pré-processamento dos ficheiros de uma transformação antes de ser enviada por HTTP verificadas na versão 4.1.0GA. Estes problemas foram discutidos com os *developers* da *Pentaho* de forma a arranjar uma resolução, resolução essa que ficou agendada para próxima *release* 4.3.0GA.

Uma vez inspecionado o código, a API original foi dividida para se adaptar às nossas necessidades consoante o levantamento de requisitos abordado na secção 7.2.2. Os *bugs* e limitações encontrados no lado do *Carte* foram resolvidos de forma semelhante às funcionalidades de extracção de variáveis discutidas anteriormente. O pré-processamento dos ficheiros XML, nomeadamente o tratamento de referências para caminhos locais de ficheiros, o tratamento de objectos partilhados entre transformações e a criação dinâmica da configuração de execução passou a ser feita no próprio *SiagETL* com base em caminhos *XPath*[54] parametrizáveis. Desta forma, numa possível futura actualização das bibliotecas da *Pentaho* tanto se podia adaptar o código já feito alterando os parâmetros *XPath*, como se podia usar directamente as bibliotecas corrigidas através da antiga API que foi implementada inicialmente no *SiagETL* e posta para já de parte face a estas dificuldades.

7.2.3 Calendarização de Execução de Transformações ETL

Para efeitos de calendarização de transformações ETL foi aproveitada a estrutura já existente de *Business Process Management*[62] no SIAG implementada sobre JBPM[28]. Isto deve-se ao facto de que o próprio JBPM já estar parcialmente preparado para processos com acções de *scheduling*, evitando assim uma implementação adicional de uma plataforma de *scheduling* aparte que abrangesse todo o sistema SIAG de forma genérica.

O processo foi elaborado conforme o diagrama presente na figura 7.3. Para a implementação foi necessário a criação de um novo componente de JBPM desenvolvido em paralelo por um membro da equipa de desenvolvimento, o *SleepScheduler*, com suporte a calendarizações complexas com vários parâmetros. Estas calendarizações são criadas por uma interface genérica de *scheduling* também construída nesta fase do projecto e demonstrada na secção seguinte. Até à data os processos de BPM presentes no SIAG apenas tiveram a necessidade de componentes temporizadores que eram activados apenas após um intervalo de tempo fixo, *hard-coded* na própria definição XML do processo.

O novo componente funciona sobre uma *string JSON*[14] que encapsula os vários parâmetros de calendarização devolvidos pelo ecrã criado através de uma variável dentro do processo. A nível da especificação XML do processo, o componente necessita da definição de dois fluxos de saída: uma saída responsável pela acção a efectuar na calendarização (que neste caso irá ser a realização de um pedido JMS ao SIAG para executar uma transformação) e uma saída responsável pelo fim da calendarização (que neste caso irá ser uma saída directa para o final do processo).

Após o *SleepScheduler* desencapsular todos os parâmetros da *string JSON* providenciada, o componente faz cálculos de tempo inteligentes para actuar de forma semelhante a um temporizador normal com alguns extras:

- Primeiro verifica quanto tempo falta para o início da calendarização especificada e espera esse tempo necessário.
- Quando é reactivado o temporizador, o componente faz o cálculo do próximo intervalo de tempo (consoante os restantes parâmetros) que necessita de estar em espera.
- Sempre que o temporizador interno do componente é reactivado consoante a calendarização, o componente segue para o próximo fluxo válido do processo.
- Quando a data limite da calendarização é atingida (se for especificada, uma vez que não é obrigatória), o próprio componente escolhe o fluxo alternativo de saída da rotina de calendarização.

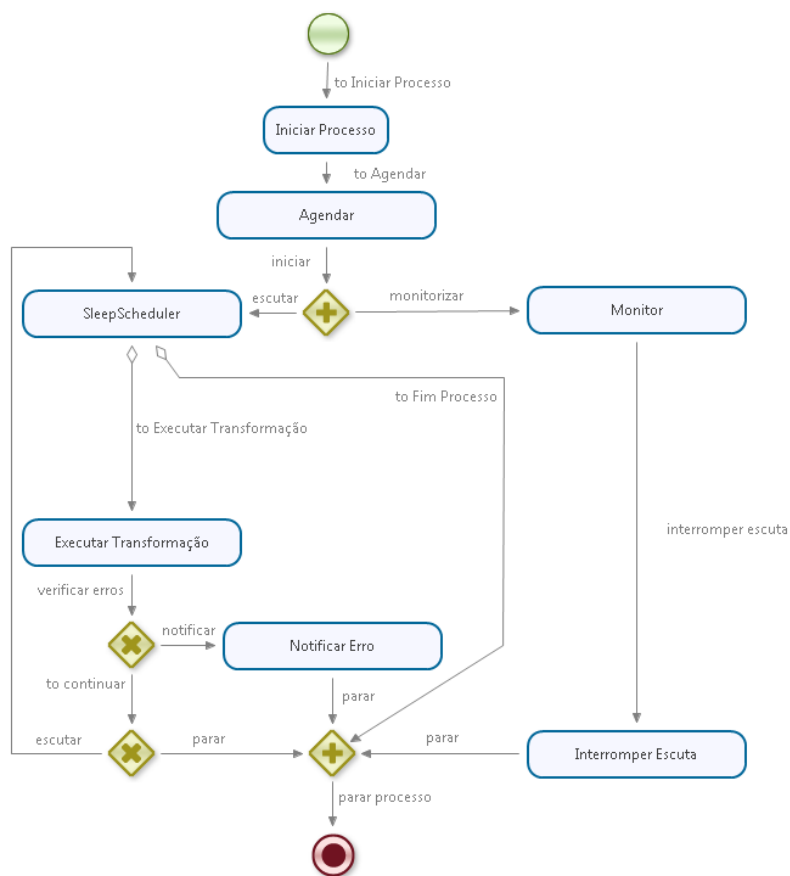


Figura 7.3: Modelo BPM do processo *ETLScheduler*

7.2.4 Criação das interfaces GWT

Para a criação das interfaces do *SiagETL*, uma vez que se tinha alguma flexibilidade de escolha, decidiu-se tentar uma abordagem semelhante à filosofia de organização da *Pentaho* no que toca à tentativa de maximizar a integração das diferentes componentes de BI de forma a que a sua utilização seja facilitada na perspectiva dos utilizadores. Com esse objectivo em mente, começou-se por desenvolver o ecrã de gravação de transformações ETL, permitindo as funcionalidades anteriormente descritas de extracção de variáveis, associar um conjunto de ficheiros criados pelo *Spoon* a uma única transformação, etc.

Este ecrã isolado foi depois integrado dentro de outro ecrã com um componente denominado no âmbito do sistema SIAG por lista de *output* (situado no lado esquerdo da figura 7.4). Este componente permite a gestão dos diferentes registos de uma determinada tabela da base de dados numa perspectiva alto-nível, onde o utilizador personaliza os campos que deseja ver, a ordenação dos registos, aplicação de vários filtros, assim como operações básicas de remoção de registos, etc. Esta extensão do uso das listas de *output* permitem aos utilizadores a criação e edição directa de transformações ETL no SIAG no mesmo ecrã, ao mesmo tempo que podem gerir outros registos de transformações ETL já gravadas no sistema.

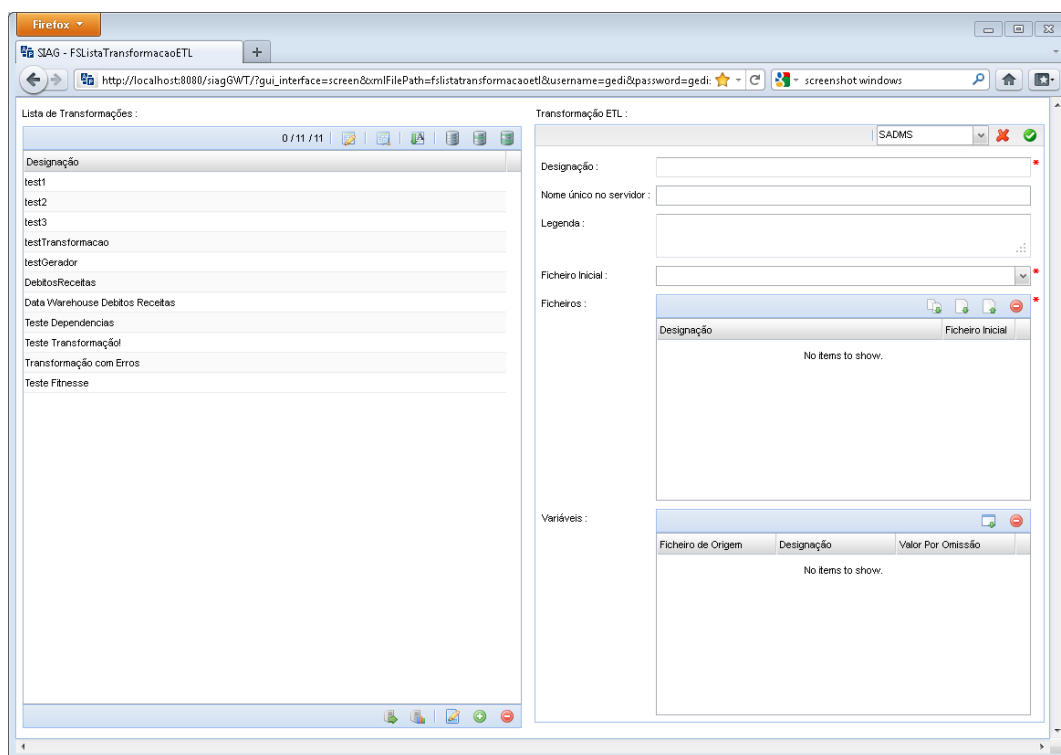


Figura 7.4: Interface de gestão de Transformações ETL no sistema SIAG-AP

Em adição da introdução deste novo componente, foi também introduzido uma *toolbar* com acesso directo a um *dashboard* de gestão de transformações no servidor *Carte* (demonstrado na figura 7.5), assim como um botão de envio das transformações ETL

selecionadas para o *Carte*. Dentro deste *dashboard* pode-se monitorizar o servidor, podendo executar uma ou várias transformações ETL ao mesmo tempo, obter o estado detalhado de uma execução, interromper uma execução que esteja a decorrer e ainda remover transformações directamente no *Carte*.

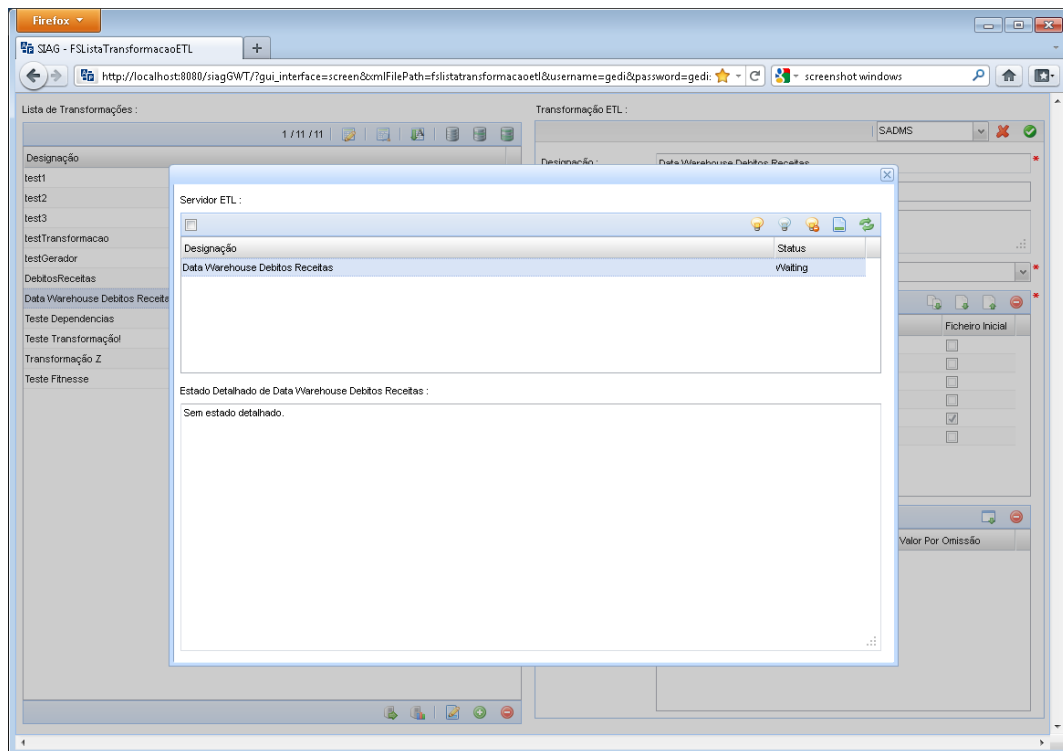


Figura 7.5: *Dashboard* de acesso ao servidor *Carte*

Capítulo 8

Conclusão

Neste capítulo irá-se abordar alguns aspectos interessantes do projecto, tais como a discussão de funcionalidades futuras que poderiam vir a ser integradas no projecto e uma breve conclusão do trabalho efectuado ao longo dos nove meses de estágio na GEDI.

8.1 Trabalho Futuro

Tendo em perspectiva o estado actual do módulo *SiagBI* e a sua integração com o sistema SIAG, assim como as lições aprendidas através da metodologia do *Agile Business Intelligence* introduzida pela *Pentaho*, creio que o primeiro grande passo será a passagem das restantes interfaces das funcionalidades de BI para a nova plataforma de interfaces GWT — nomeadamente o explorador de cubos e os ecrãs de criação de indicadores e gráficos sobre os dados de um cubo. Isto permitirá, tendo em conta as funcionalidades da plataforma *SiagGWT* de integração recursiva de formulários no mesmo ecrã, novas maneiras de integrar as funcionalidades do módulo de BI, assim como abrir as portas ao mercado *mobile* (os ecrãs de GWT são compatíveis com *browsers* de dispositivos móveis). Neste novo paradigma, seria fácil agregar as funcionalidades de BPM num único *dashboard* universal, aumentando a produtividade do utilizador final e a simplicidade de uso com a redução do número de ecrãs necessários. Numa perspectiva ideal, este *dashboard* universal, à semelhança dos *dashboards* de dados de um cubo e o módulo de desenho personalizado de ecrãs GWT, poderia vir a ser personalizado pelo próprio utilizador de acordo com as suas necessidades específicas.

No entanto a maior possibilidade de expansão futura recai no módulo *SiagETL*. Devido ao tempo limitado de desenvolvimento do projecto, o âmbito da implementação foi limitado à execução de ficheiros de transformação já preparados pela interface do *Spoon*. No entanto, com o amadurecimento da plataforma *SiagGWT* em relação a componentes de desenho de grafos, a criação de transformações ETL poderia ser feita através das próprias interfaces GWT. O controlo directo sobre a especificação dos componentes que compõe o modelo de uma transformação ETL implicaria uma redução de complexidade

da construção do mesmo para o utilizador final, abstraindo parametrizações que foram tomadas como desnecessárias aos utilizadores do SIAG. O próprio desenho dos componentes poderia também ser melhorado na organização de campos, ficando mais apelativo.

Face ao amadurecimento da plataforma *SiagGWT* referido anteriormente, poderia também ser desenvolvido uma nova funcionalidade de desenho gráfico de *data marts*, onde o utilizador desenhava interactivamente o modelo de dados pretendido, especificando uma fonte de dados. Os módulos de base de dados implementados, presentes no *SiagBI*, poderiam ser expandidos para conseguir gerar *statements SQL* para criação de tabelas, índices e chaves para dar suporte ao novo módulo de construção de *data marts*. Desta forma era reforçado o uso de ambos os módulos *SiagBI* e *SiagETL* proximamente como uma suite de BPM completa, permitindo desde a criação e manutenção de *data warehouses* à exploração de dados de cubos OLAP, recolha de métricas e criação de gráficos sobre esses mesmos *data warehouses*.

Outra funcionalidade futura seria usufruir das novas funcionalidades de envio de mensagens JMS do *Kettle* para fazer chamadas directas ao SIAG num dos passos da transformação. Isto permitiria novas formas de integração com o sistema, abrindo a possibilidade de criação ou importação de registos de base de dados re-utilizando, por exemplo, as validações já implementadas na camada de negócio do SIAG. Desta forma, o uso do ETL para importações directas na base de dados seria mais directa e simplificada, não exigindo a recriação de validações ou outros algoritmos necessários já existentes através dos modelos de processos de ETL.

Por fim, seria interessante à medida que o novo módulo de ETL vai ganhando aderência por parte dos utilizadores, a disponibilização de transformações que viessem incluídas no sistema SIAG para utilização dos utilizadores, à semelhança dos ecrãs GWT que são importados juntamente com sistema SIAG. Desta forma poderia-se abstrair operações complexas comuns em sub-transformações de sistema que poderiam vir a ser utilizadas como blocos de construção já prontos dentro de processos de ETL mais complexos. No entanto este tipo de transformações teriam de ser modeladas face a requisitos comuns que se encontrassem nos padrões de utilização de processos de ETL modelados pelos utilizadores do SIAG.

8.2 Conclusão

O estágio durou aproximadamente nove meses, respeitando assim os prazos estabelecidos na fase de planeamento do projecto, uma vez que foi possível cumprir todos os objectivos propostos pela organização. Entre estes objectivos, pode-se destacar a criação do novo módulo de *Business Intelligence* flexível, escalável e com melhor capacidade de manutenção, podendo até mesmo ser integrado num contexto fora do sistema SIAG-AP devido à independência introduzida no módulo. Foi também dado início à passagem das

antigas interfaces do SIAG respectivas às funcionalidades de BI para a nova plataforma baseada em GWT, alinhando cada vez mais o novo módulo para a nova arquitectura escalável discutida anteriormente neste documento. Foram também cumpridos os objectivos de introdução de processos de *Extract, Transform and Load* no SIAG através do novo módulo de ETL, capaz de executar processos de ETL complexos modelados através da interface do *Spoon*, uma interface apelativa aos utilizadores da área de negócio que usufruem do sistema SIAG como sistema ERP nas suas organizações. Foi também possível parametrizar parcialmente as transformações ETL carregadas no SIAG através das novas interfaces criadas, permitindo uma maior flexibilidade na construção de modelos de processos de ETL e melhorando a produtividade do utilizador final ao reduzir o número de passos para a actualização de uma determinada transformação sem ter de alterar todos os seus ficheiros respectivos carregados no SIAG.

Estes novos módulos independentes, juntamente com o desenvolvimento paralelo e amadurecimento da plataforma *SiagGWT*, possibilitam uma evolução favorável do SIAG-AP no que toca as tendências do mercado de BPM analisadas na fase de investigação do projecto, possibilitando no futuro uma integração cada vez mais próxima e coesa dos vários componentes de sistema numa perspectiva única, personalizável pelo utilizador. A criação de *dashboards* com agregados de ecrãs seleccionados pelos utilizadores permitirão uma forma cada vez mais ágil de acesso, processamento e análise da informação num só ecrã com todas as funcionalidades integradas, aumentando drasticamente a performance de um utilizador comum que necessite constantemente de aceder aos diferentes ecrãs de BI, dispersos no actual sistema.

Em termos de dificuldades encontradas, pode-se destacar três pontos fulcrais. O primeiro problema passou pela refactorização do módulo de BI para conseguir atingir a independência tão desejada da camada de persistência do SIAG, uma vez que as *queries SQL* necessárias aos ficheiros XML do *Mondrian* passaram a ser geradas manualmente através de algoritmos complexos implementados durante o projecto. Juntamente a este passo, também se pode referir a adição de suporte a ligações *many-to-one* no processamento de caminhos de dimensões durante a geração desse XML, que levou também a uma re-estruturação complexa do algoritmo criado, assim como à da semântica usada para a especificação de caminhos entre tabelas em todo o módulo.

O último ponto que merece ser referido nas dificuldades encontradas foi a passagem do pré-processamento de ficheiros de transformação do servidor *Carte* para o próprio processamento dentro do módulo *SiagETL*, exigindo uma inspecção cuidada do código fonte disponibilizado, assim como vários testes funcionais sobre o servidor para a implementação de uma solução simples, genérica e flexível, capaz de ser facilmente parametrizável face a mudanças nas versões seguintes do *Pentaho Data Integration*[47].

Para terminar, posso afirmar em termos pessoais este projecto permitiu-me desenvolver fortes competências para além da minha formação académica em Engenharia de

Software na Faculdade de Ciências da Universidade de Lisboa. Entre estas competências pode-se destacar o domínio das tecnologias *Spring*, *Hibernate*, *Mondrian* e *Pentaho Data Integration* e domínio na programação web sobre a plataforma J2EE. Posso também destacar as grandes melhorias em relação à programação em java e na compreensão geral de padrões de desenho de software, principalmente no desenvolvimento do módulo *data-base* incorporado no *SiagBI*. A criação deste módulo exigiu a criação de uma abstração no acesso à informação de várias bases de dados, ao mesmo tempo que foi necessário lidar com os pormenores da implementação de baixo-nível para cada tipo de base de dados usado, reforçando a minha experiência de trabalho em cada uma delas.

Bibliografia

- [1] Matt Casters. *Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration*. Wiley, 2010.
- [2] Comparing Talend Open Studio and Pentaho Data Integration. <http://churriwif.wordpress.com/2010/06/01/comparing-talend-open-studio-and-pentaho-data-integration-kettle/>.
- [3] Dan Mihai Ile. Desenvolvimento de um RAD para aplicações WEB (*Rapid Application Development*). Master's thesis, Faculdade de Ciências da Universidade de Lisboa, 2009. A base do projecto SiagGWT.
- [4] Data Warehouse Concepts! http://etl-tools.info/en/bi/datawarehouse_concepts.htm.
- [5] Documentação da classe java DatabaseMetaData. <http://download.oracle.com/javase/6/docs/api/java/sql/DatabaseMetaData.html>.
- [6] Engine-based vs. code-generating ETL tools. <http://searchoracle.techtarget.com/answer/Engine-based-vs-code-generating-ETL-tools>.
- [7] Firebird Database. <http://www.firebirdsql.org/>.
- [8] Gabinete de Estudos e Divulgação Informática. <http://www.gedi.pt/portal/default/Empresa>.
- [9] Gartner Group. <http://www.gartner.com/technology/home.jsp>.
- [10] Hibernate - JBoss Community. <http://www.hibernate.org/>.
- [11] Gregor Hohpe. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2003.
- [12] IBM. The New Voice of the CIO: Insights from the Global Chief Information Officer Study. Technical report, IBM, 2009.
- [13] Java Blueprints: Model-View-Controller. <http://java.sun.com/blueprints/patterns/MVC-detailed.html>.

- [14] Javascript Object Notation. <http://www.json.org/>.
- [15] JBoss Application Server. <http://www.jboss.org/jbossas/>.
- [16] Ralph Kimball. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*. Wiley, 2002.
- [17] Mantis Bug Tracker. <http://www.mantisbt.org/>.
- [18] Mozilla Rhino Project Home. <http://www.mozilla.org/rhino/>.
- [19] Open Source BI: Pentaho Rules! <http://www.ibridge.be/?p=178>.
- [20] Oracle: JavaServer Pages Technology. <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>.
- [21] Pentaho Business Intelligence Suite. http://www.pentaho.com/products/bi_suite/?hp=y.
- [22] Pentaho Data Integration VS. Talend. <http://www.ibridge.be/?p=150>.
- [23] Pentaho: Demos, Case Studies, White Papers, and Market Insights. <http://www.pentaho.com/products/demos/showNtell.php?tab=demos>.
- [24] Pentaho Weka Project. <http://weka.pentaho.com/>.
- [25] Pentaho Wiki: BI Server Architecture and Philosophy. <http://wiki.pentaho.com/display/PEOpen/BI+Server+architecture+and+philosophy>.
- [26] Pentaho Wiki: The Pentaho Administration Console. <http://wiki.pentaho.com/display/ServerDoc2x/The+Pentaho+Administration+Console>.
- [27] Pentaho Wiki: Welcome to Agile Business Intelligence. <http://wiki.pentaho.com/display/AGILEBI/Welcome+to+Agile+Business+Intelligence>.
- [28] Página oficial do JBPM. <http://www.jboss.org/jbpm>.
- [29] Página oficial do Jetty. <http://jetty.codehaus.org/jetty/>.
- [30] Página oficial do JPivot. <http://jpivot.sourceforge.net/>.
- [31] Página oficial do Mondrian. <http://mondrian.pentaho.com/>.
- [32] Review of the ETL Vendor Comparison Report. <http://it.toolbox.com/blogs/infosphere/my-review-of-the-etl-vendor-comparison-report-from-wwwetltoolcom-33310>.
- [33] Sistema Integrado de Apoio à Gestão. <http://www.gedi.pt/portal/default/O+Sistema+SIAG>.

- [34] Sistema Integrado de Apoio à Gestão da Administração Pública. <http://www.gedi.pt/portal/default/O+SIAG-AP>.
- [35] Site Oficial da Empresa IBM. <http://www.ibm.pt>.
- [36] Site oficial da Pentaho. <http://www.pentaho.com>.
- [37] Site oficial da SourceForge. <http://sourceforge.net/>.
- [38] Site oficial da Spring Framework. <http://www.springsource.org/about>.
- [39] Site oficial das bases de dados MySQL. <http://www.mysql.com/>.
- [40] Site oficial das bases de dados Oracle. <http://www.oracle.com/us/products/database/index.html>.
- [41] Site oficial das bases de dados SQL Server. <http://www.microsoft.com/sqlserver/en/us/default.aspx>.
- [42] Site Oficial do ActiveMQ. <http://activemq.apache.org/>.
- [43] Site Oficial do CloverETL. <http://www.cloveretl.com/>.
- [44] Site Oficial do Crystal Reports. <http://www.crystalreports.com>.
- [45] Site Oficial do Eclipse. <http://www.eclipse.org>.
- [46] Site oficial do notação PBMN. <http://www.bpmn.org/>.
- [47] Site Oficial do Pentaho Data Integration / Kettle. <http://kettle.pentaho.com/>.
- [48] Site Oficial do projecto Java Single Sign-On. <http://www.josso.org/>.
- [49] Site Oficial do Struts. <http://struts.apache.org/>.
- [50] Site Oficial do Talend Open Studio. <http://www.talend.com/products-data-integration/talend-open-studio.php>.
- [51] Site Oficial dos projectos da empresa Enhydra. <http://www.together.at/prod/workflow/tws>.
- [52] Tips for the ETL Tool Evaluation and Comparison Process. <http://searchoracle.techtarget.com/answer/Tips-for-the-ETL-tool-evaluation-and-comparison-process>.
- [53] W3Schools: SQL Joins. http://www.w3schools.com/sql/sql_join.asp.
- [54] W3Schools: XPath. <http://www.w3schools.com/xpath/default.asp>.
- [55] What is XMLA. <http://news.xmlforanalysis.com/what-is-xmla.html>.

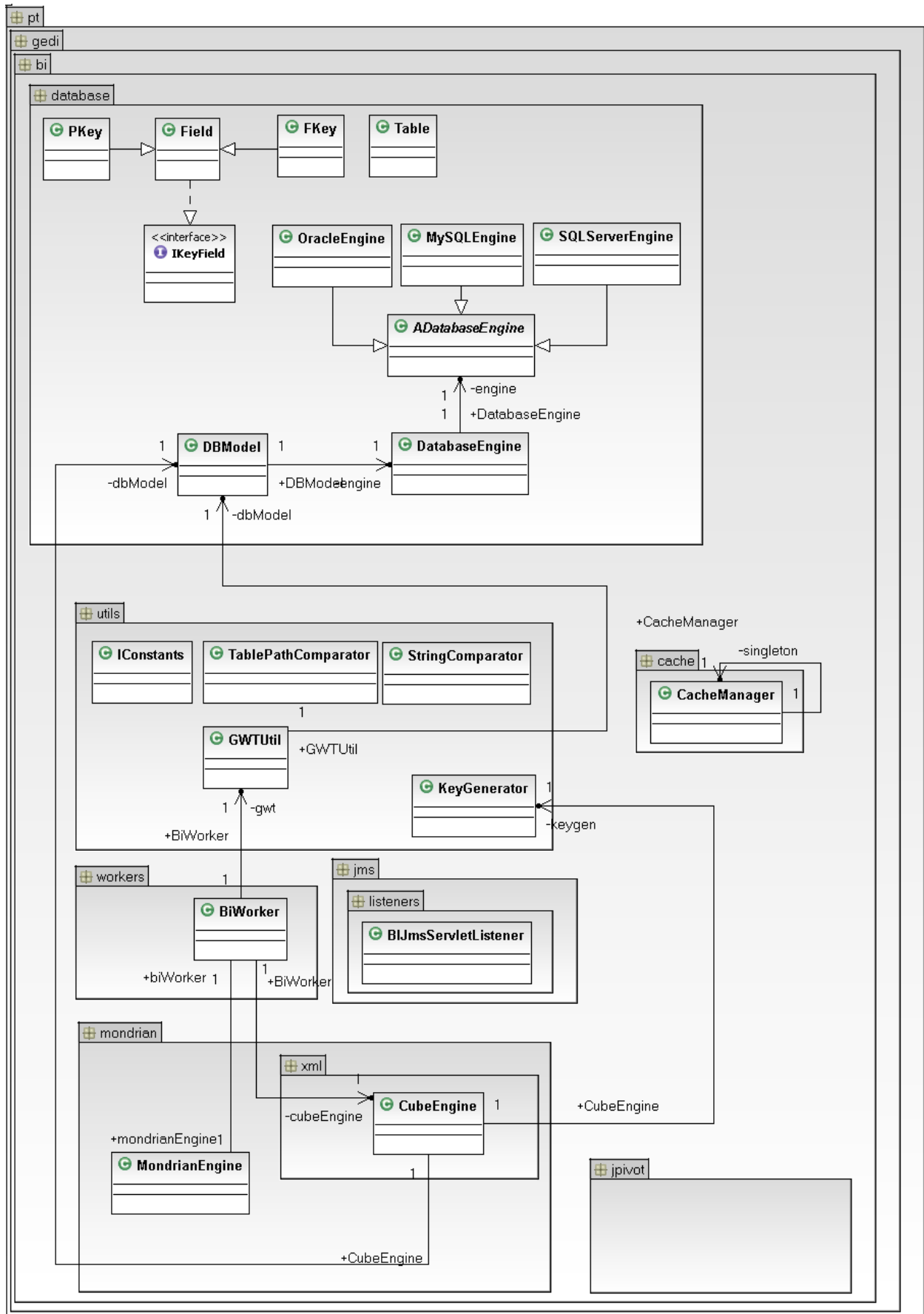
- [56] What parameters should be considered to compare the top five ETL tools?
<http://www.information-management.com/news/1044005-1.html>.
- [57] WhatIs: Slice And Dice. *http://whatis.techtarget.com/definition/0,,sid9_gci212997,00.html.*
- [58] Wikipedia: Bill Inmon. *http://en.wikipedia.org/wiki/Bill_Inmon.*
- [59] Wikipedia: Definição de Business-Driven Development.
http://en.wikipedia.org/wiki/Business-driven_development.
- [60] Wikipedia: Definição de Business Intelligence.
http://en.wikipedia.org/wiki/Business_intelligence.
- [61] Wikipedia: Definição de Business Performance Management.
http://en.wikipedia.org/wiki/Business_performance_management.
- [62] Wikipedia: Definição de Business Process Management.
http://en.wikipedia.org/wiki/Business_process_management.
- [63] Wikipedia: Definição de Cluster. *<http://pt.wikipedia.org/wiki/Cluster>.*
- [64] Wikipedia: Definição de Cubo OLAP. *http://en.wikipedia.org/wiki/OLAP_cube.*
- [65] Wikipedia: Definição de Customer Relationship Management.
http://pt.wikipedia.org/wiki/Customer_relationship_management.
- [66] Wikipedia: Definição de Dashboard. *<http://en.wikipedia.org/wiki/Dashboard>.*
- [67] Wikipedia: Definição de Data Access Object.
http://en.wikipedia.org/wiki/Data_access_object.
- [68] Wikipedia: Definição de Data Dictionary. *http://en.wikipedia.org/wiki/Data_dictionary.*
- [69] Wikipedia: Definição de Data Mart. *http://en.wikipedia.org/wiki/Data_mart.*
- [70] Wikipedia: Definição de Data Mining. *http://en.wikipedia.org/wiki/Data_mining.*
- [71] Wikipedia: Definição de Data Quality. *http://en.wikipedia.org/wiki/Data_quality.*
- [72] Wikipedia: Definição de Data Warehouse. *http://en.wikipedia.org/wiki/Data_warehouse.*
- [73] Wikipedia: Definição de Drill-Down. *http://en.wikipedia.org/wiki/Drill_down.*
- [74] Wikipedia: Definição de Enterprise Application Integration.
http://en.wikipedia.org/wiki/Enterprise_application_integration.
- [75] Wikipedia: Definição de Enterprise Resource Planning.
http://en.wikipedia.org/wiki/Enterprise_resource_planning.

- [76] Wikipedia: Definição de Extract, Transform and Load. http://en.wikipedia.org/wiki/Extract,_transform,_load.
- [77] Wikipedia: Definição de HOLAP. <http://en.wikipedia.org/wiki/HOLAP>.
- [78] Wikipedia: Definição de Java Database Connectivity. <http://pt.wikipedia.org/wiki/JDBC>.
- [79] Wikipedia: Definição de Java Message System. http://en.wikipedia.org/wiki/Java_Message_Service.
- [80] Wikipedia: Definição de Key Performance Indicators. http://en.wikipedia.org/wiki/Performance_indicator.
- [81] Wikipedia: Definição de MDX. http://en.wikipedia.org/wiki/MultiDimensional_eXpressions.
- [82] Wikipedia: Definição de MOLAP. <http://en.wikipedia.org/wiki/MOLAP>.
- [83] Wikipedia: Definição de Online Analytical Processing. http://en.wikipedia.org/wiki/Online_analytical_processing.
- [84] Wikipedia: Definição de Plain Old Java Object. http://en.wikipedia.org/wiki/Plain_Old_Java_Object.
- [85] Wikipedia: Definição de Rapid Application Development. http://en.wikipedia.org/wiki/Rapid_application_development.
- [86] Wikipedia: Definição de ROLAP. <http://en.wikipedia.org/wiki/ROLAP>.
- [87] Wikipedia: Definição de SNMP. http://pt.wikipedia.org/wiki/Simple_Network_Management_Protocol.
- [88] Wikipedia: Definição de Swing no contexto da linguagem Java. [http://en.wikipedia.org/wiki/Swing_\(Java\)](http://en.wikipedia.org/wiki/Swing_(Java)).
- [89] Wikipedia: Definição de Workflow Engine. http://en.wikipedia.org/wiki/Workflow_engine.
- [90] Wikipedia: Definição de XPDL. <http://en.wikipedia.org/wiki/XPDL>.
- [91] Wikipedia: Definição do padrão Abstract Factory. http://en.wikipedia.org/wiki/Abstract_factory_pattern.
- [92] Wikipedia: Definição do padrão Facade. http://en.wikipedia.org/wiki/Facade_pattern.
- [93] Wikipedia: Definição do padrão Message Broker. http://en.wikipedia.org/wiki/Message_broker.

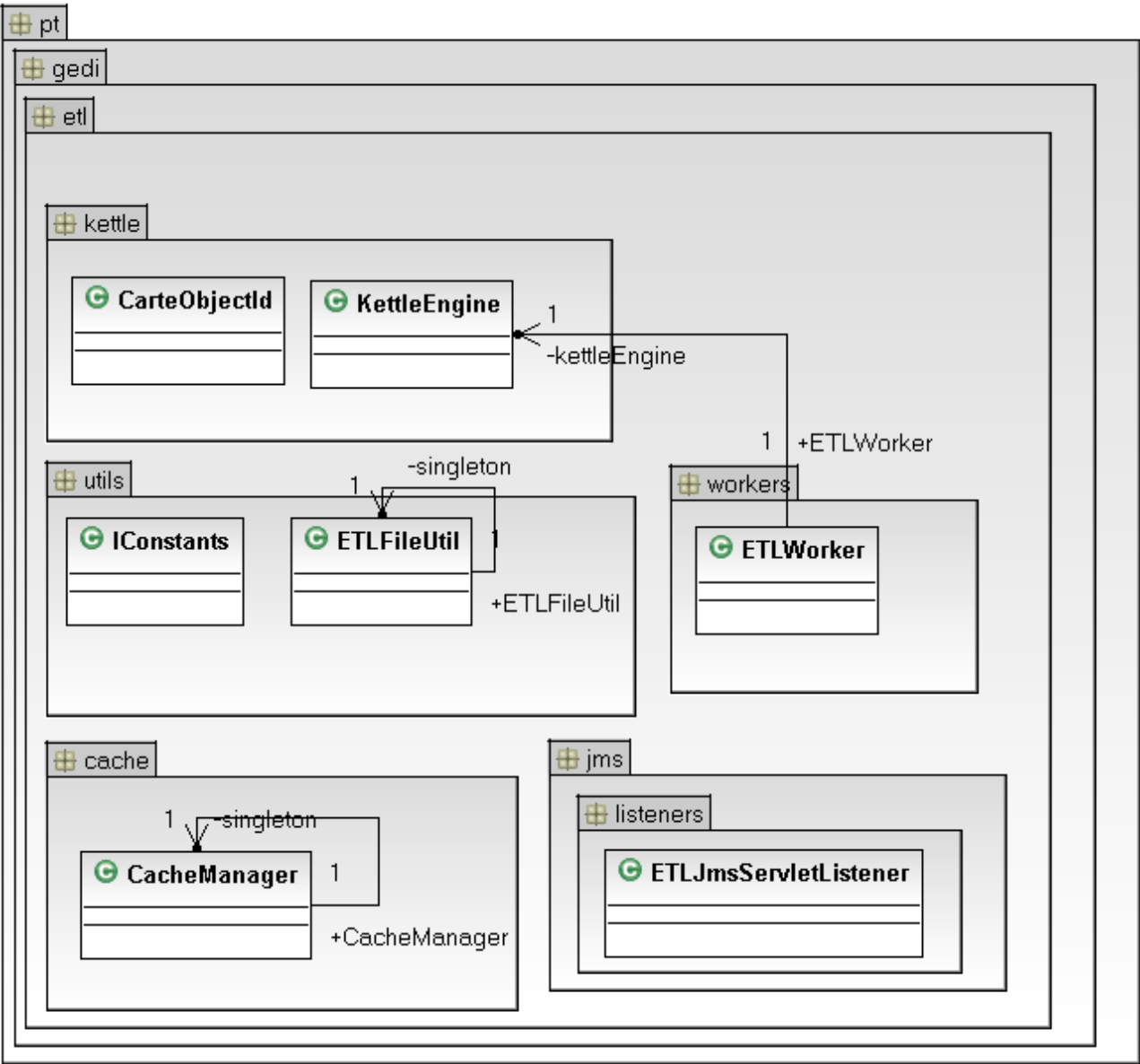
[94] Wikipedia: Definição do padrão Singleton.
http://en.wikipedia.org/wiki/Singleton_pattern.

[95] Wikipedia: Ralph Kimball. *http://en.wikipedia.org/wiki/Ralph_Kimball.*

Anexo A – Modelo UML do módulo SiagBI



Anexo B – Modelo UML do módulo SiagETL



Anexo C – Análise de Dados (PEI)



ANÁLISE DE DADOS (PEI)

REFERÊNCIA: INVESTIGAÇÃO DA PENTAHO DATA INTEGRATION COMPONENT, TECNOLOGIAS DE DATA WAREHOUSE E ETL

AUTOR:	JOÃO NATÁLIO	DATA:	10/09/2010	REVISOR:	NUNO RODRIGUES	DATA:	DD/MM/AAAA	VERSÃO:	(1.0)
VERSÃO INICIAL									

DOCUMENTO PARA USO INTERNO E EXCLUSIVO DA GEDI

ÍNDICE

1	-INTRODUÇÃO.....	3
2	-CONCEITOS PRINCIPAIS.....	4
2.1	-BUSINESS INTELLIGENCE.....	4
2.2	-BUSINESS PERFORMANCE MANAGEMENT.....	4
2.3	-OLAP.....	4
2.4	-DATA WAREHOUSE.....	5
2.5	-DATA MART.....	5
2.6	-EXTRACT, TRANSFORM AND LOAD (ETL).....	6
2.7	-ESQUEMA EM ESTRELA.....	6
2.8	-OPERATIONAL DATA STORES (ODS).....	7
3	-INVESTIGAÇÃO.....	8
3.1	-DATA WAREHOUSE.....	8
3.2	-PROCESSO DE ETL.....	12
4	-PENTAHO BUSINESS INTELLIGENCE SUITE.....	13
4.1	-PENTAHO SERVER.....	15
4.2	-PENTAHO ENTERPRISE CONSOLE.....	18
4.3	-PENTAHO DESIGN STUDIO.....	20
4.4	-PENTAHO METADATA EDITOR.....	22
4.5	-PENTAHO REPORT DESIGNER.....	24
4.6	-PENTAHO SCHEMA WORKBENCH.....	26
4.7	-PENTAHO AGGREGATION DESIGNER.....	27
4.8	-PENTAHO ANALYSER.....	28
4.9	-PENTAHO DATA INTEGRATION (KETTLER).....	32
4.10	-PRODUTOS OPEN SOURCE DA PENTAHO.....	35
5	-MANUAL DE UTILIZAÇÃO DO PENTAHO DATA INTEGRATION.....	36
5.1	-ECLIPSE SETUP.....	36
5.2	-CASOS DE USO – CRIAR UMA TRANSFORMAÇÃO E EXECUTÁ-LA.....	36
5.3	-CASOS DE USO – CRIAR UMA TAREFA E EXECUTÁ-LA.....	42
5.4	-CASOS DE USO – EXECUTAR UMA TRANSFORMAÇÃO NO JAVA API.....	44
5.5	-CASOS DE USO – EXECUTAR UMA TAREFA NO JAVA API.....	45
5.6	-CASOS DE USO – CONFIGURAR E LANÇAR O SERVIDOR CARTE MANUALMENTE.....	46
5.7	-CASOS DE USO – EXECUTAR UMA TRANSFORMAÇÃO REMOTAMENTE NO CARTE.....	47
6	-OUTRAS FRAMEWORKS ETL OPEN SOURCE.....	49
6.1	-TALEND OPEN STUDIO.....	49
6.2	-CLOVERETL.....	52



G E D I

7 -	CONCLUSÃO	55
8 -	REFERÊNCIAS IMPORTANTES	56



1 - INTRODUÇÃO

O objectivo deste relatório consiste em investigar metodologias actuais de desenvolvimento de data warehouses e as tecnologias relacionadas com este problema, onde iremos destacar um conjunto de produtos de business intelligence open source a nível de arquitectura e funcionalidades, o *Pentaho Business Intelligence Suite*.

Dentro desta suite iremos então tentar averiguar quais os componentes dotados de funcionalidades de data warehousing e ETL, comparando também com outras frameworks java open source semelhantes. Esta comparação servirá para averiguar qual a melhor opção dentro das ofertas presentes no mercado para a podermos integrar no nosso projecto.

A nossa meta principal é, em conjunto com este relatório e os seus resultados, a criação de uma framework focada na monitorização e optimização da performance de negócio dentro de uma organização. Esta framework será um recurso importante de apoio à decisão, através de análise e automatização de processos de negócio, assim como a análise de métricas e indicadores chave com o propósito de auxiliar gestores em tomadas de decisões que poderão influenciar de forma positiva a performance das suas organizações.

A framework, uma vez finalizada, será então integrada no nosso sistema SIAG-AP (Sistema Integrado de Apoio à Gestão para Administração Pública) e terá de funcionar de forma autónoma para que possa mais tarde ser integrada com sistemas de ERP de outras empresas.



2 - CONCEITOS PRINCIPAIS

Relacionado com o projecto existem diversos conceitos necessários para obter um melhor entendimento do mesmo no seu todo. Nesta secção faremos uma breve introdução a cada um destes conceitos essenciais para servir de base ao resto da investigação.

2.1 - BUSINESS INTELLIGENCE

Refere-se ao conjunto de técnicas baseadas nas tecnologias de informação usadas para recolher, organizar, analisar, compartilhar e monitorizar informação relevante e necessária à gestão eficiente de negócio. É usada principalmente como ferramenta para tomar decisões mais informadas e precisas de negócio.

Este conceito também está fortemente ligado às tecnologias de data warehouse, já que é através deste que são extraídos e agregados dados relevantes que irão servir como suporte a todas as operações e metodologias de business intelligence.

2.2 - BUSINESS PERFORMANCE MANAGEMENT

Este termo é usado para definir o conjunto de processos de análise e gestão que permitem a monitorização e gestão da performance de um organização em relação a um ou mais objectivos estratégicos pré-determinados. É focado em três actividades principais:

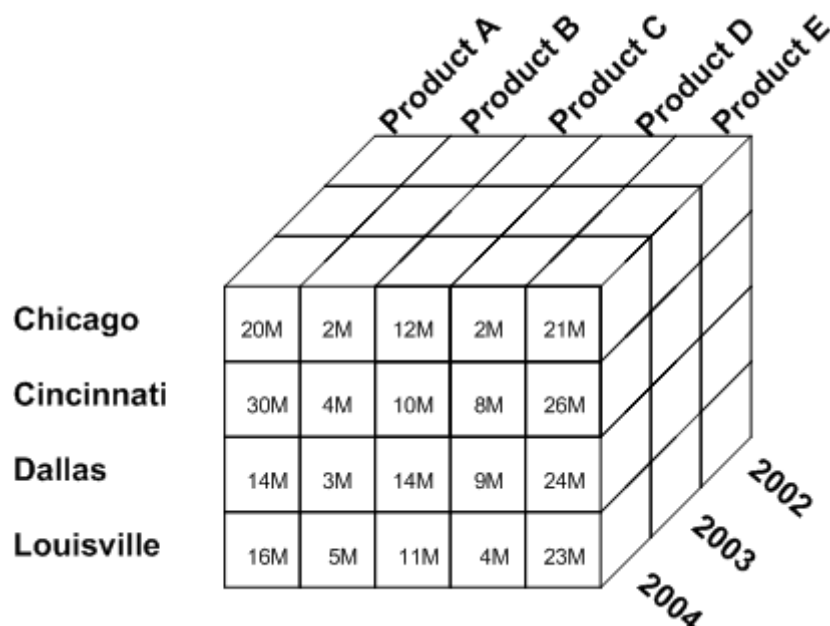
- A selecção de objectivos, consoante a estratégia da empresa.
- A consolidação de informação relevante ao progresso da organização em relação a esses objectivos.
- As intervenções feitas por gestores em relação à informação recolhida, tendo em vista o melhoramento da performance da organização.

Ao longo do tempo foram sendo desenvolvidas ferramentas de suporte ao trabalho do BPM, entre estas destacam-se aquelas que iremos investigar para integração no projecto, nomeadamente o OLAP para efectuar análise de dados e as já referidas data warehouses para agregar informação relevante.

2.3 - OLAP

Também referido como *online analytical processing*, é a abordagem usada para responder prontamente a queries analíticas multi-dimensionais complexas. As bases de dados são configuradas para funcionar em conjunto com o OLAP, recorrendo a um modelo de dados multidimensional de forma a manter uma performance adequada à quantidade bruta de informação necessária para realizar processos de BI, assim como as várias dimensões em que é necessário analisá-la.

O núcleo de um sistema OLAP consiste num cubo multidimensional como demonstra a figura:



Este cubo contém factos numéricos recolhidos de bases de dados relacionais designados por *medidas* (neste caso são os valores monetários presentes no esquema). Cada *medida* está associada a um conjunto de *etiquetas* ou meta-dados (por exemplo, Chicago, Cincinnati, etc. são *etiquetas* do esquema, representam cidades). As *medidas* são categorizadas em *dimensões* e é aquilo que descreve estas *etiquetas* (um exemplo de uma *dimensão* no esquema seria um corte do cubo feito no produto B, dando informações financeiras categorizadas por cidade e por ano apenas sobre esse produto).

Os resultados das queries são depois tipicamente retornados numa matriz, onde as dimensões formam as colunas e linhas da matriz e as medidas formam os valores que a preenchem.

2.4 - DATA WAREHOUSE

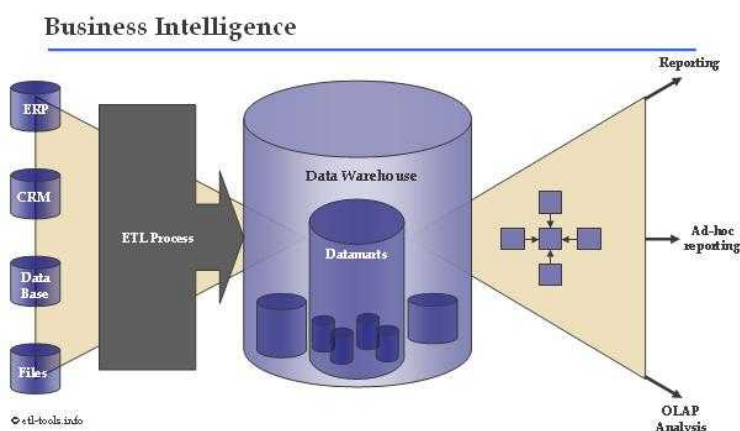
É um repositório de dados relativos a uma organização, desenhado para facilitar a recolha e análise de informação relevante e que irá de servir de suporte a outros processos dentro da organização. Neste caso focamo-nos no uso destes data warehouses como suporte à nossa framework BPM.

2.5 - DATA MART

É uma pequena parte do armazenamento organizacional presente numa empresa. São unidades de armazenamento de dados analíticos desenhados para se focarem em determinadas funções de negócio para uma comunidade específica dentro da organização. Data marts são quase sempre subsets de dados de um data warehouse, exceptuando o caso de um data warehouse desenhado numa aproximação “bottom-up”, onde este é criado através da união dos data marts.



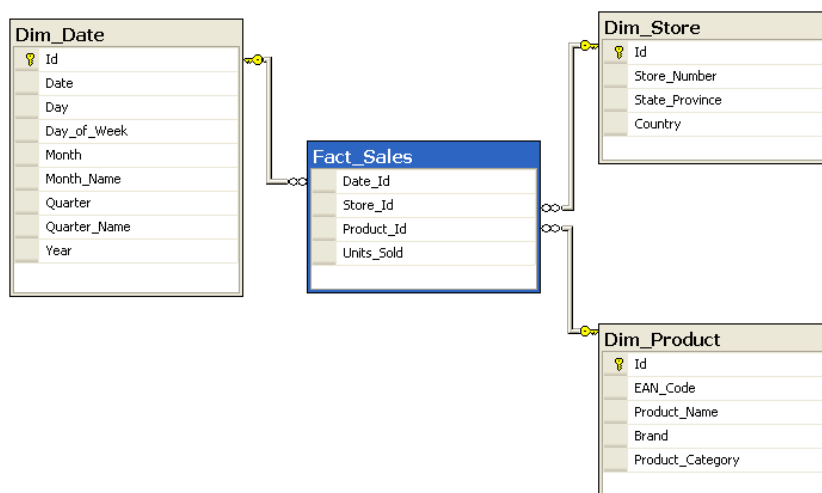
2.6 - EXTRACT, TRANSFORM AND LOAD (ETL)



É um processo relacionado com o uso de base de dados e data warehousing que envolve a extracção de dados de fontes externas, seguido da transformação destes dados para se adaptarem aos nossos requisitos de negócio e terminando com o carregamento desta informação trabalhada numa base de dados ou data warehouse. Seguidamente apresentamos um esquema simples para mostrar o contexto do processo ETL num sistema dedicado ao business intelligence:

2.7 - ESQUEMA EM ESTRELA

É um esquema de modelação de dados simplístico de um data warehouse. Consiste num conjunto pequeno de tabelas de factos, tipicamente apenas uma tabela, que serve de referência para qualquer número de tabelas de dimensões no repositório de dados. As tabelas de factos costumam armazenar dados numéricos (ligados a preços, quantidades, etc.) e depois estende o seu conhecimento sempre que necessário para uma tabela de dimensão, descritiva de um determinado atributo ligado a esses dados. Heis um exemplo de um esquema em estrela:





2.8 - OPERATIONAL DATA STORES (ODS)

É uma base de dados usada para integrar dados de múltiplas fontes, executando operações de limpeza, resolução de redundância e verificação de regras de negócio para preparação da informação. Serve de suporte de baixo nível aos repositórios de dados maiores presentes numa organização, tais como data warehouses e data marts. Apenas guarda informação atómica de forma mais limitada em termos de capacidade de armazenamento, mas actualizada mais frequentemente em comparação com um data warehouse.



3 - INVESTIGAÇÃO

Nesta secção faremos uma documentação dos principais pontos em que nos temos que focar, nomeadamente uma análise a fundo do que pode ser aproveitado da Pentaho BI Suite para ser utilizado no nosso projecto, assim como discussão de alternativas viáveis para conseguirmos implementar um ou mais Data Warehouses na framework a desenvolver.

3.1 - DATA WAREHOUSE

Um data warehouse, tal como já foi dito, trata-se de um repositório de informação dedicado e centralizado dentro duma empresa para suportar operações de BI.

Os dados presentes neste tipo de repositórios são sempre detalhados, não-voláteis (nunca são apagados ou sobrescritos), orientados a um tema e são variantes com o tempo (todas as mudanças que ocorrem sobre os dados são monitorizadas, permitindo uma análise dos mesmos sob um contexto histórico).

Uma vez que já possuímos uma base de dados dimensional pronta e integrada com um servidor OLAP, apenas temos de nos preocupar com a metodologia escolhida para o design do nosso data warehouse e, possivelmente, alguns data marts especializados.

Existem quatro metodologias de design de data warehouses:

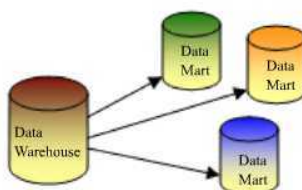
- Abordagem “top-down”, proposta por Bill Inmon.
- Abordagem “bottom-up”, proposta por Ralph Kimball.
- Abordagem híbrida.
- Abordagem federada.

Iremos então entrar em detalhe em cada uma destas abordagens para tentar perceber qual delas é que nos pode ser útil e qual é que nos pode trazer melhores vantagens dado o contexto e ambiente de desenvolvimento da nossa organização.

DESIGN “TOP-DOWN”

Neste design é o data warehouse que guarda os dados atómicos ou de transacção que são extraídos de diversas fontes de informação externas, para depois serem integradas num modelo normalizado de dados empresariais.

Os dados, após a sua extracção para um data warehouse central, são sintetizados, dimensionados e distribuídos por entre um ou mais data marts especializados, pertencendo cada um deles normalmente a um departamento dentro da organização. Sendo assim, todos os dados presentes nestes data marts são dependentes da informação retirada do data warehouse. Os dados são então acedidos em detalhe atómico ao explorarmos os níveis inferiores (dentro dos data marts), ou então são acedidos de forma sintetizada ao explorarmos os níveis de topo da estrutura (dentro do data warehouse central).





Este design tem algumas vantagens, tais como:

- Impõe uma arquitectura empresarial flexível, permitindo às organizações reestruturar os dados em qualquer número de maneiras para obedecer a possíveis novos desafios que possam surgir.
- Minimiza a presença de data marts renegados no sistema, uma vez que todos dependem directamente do data warehouse central.
- Impõe consistência e standardização sobre todos os dados do sistema e reduz redundância na informação extraída.

No entanto, este modelo não deixa de ter desvantagens na sua implementação:

- Demora mais tempo e despende de mais recursos da organização para ser implementado em comparação às restantes abordagens, nomeadamente nos requisitos iniciais de modelação.
- Torna-se difícil para os utilizadores explorar a informação em detalhe no data warehouse, partindo da informação sintetizada de um data mart.
- Pode haver necessidade à mesma de guardar dados detalhados num data mart.

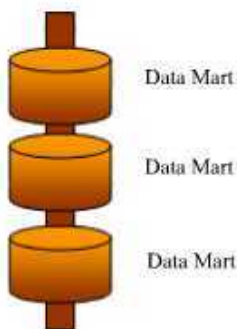
DESIGN “BOTTOM-UP”

Numa abordagem “bottom-up”, o objectivo passa por definir data marts dimensionais o mais depressa possível, tentando focar-se no desenvolvimento de designs dimensionais prontos a cumprir os requisitos do cliente.

Estes data marts, ao contrário do design “top-down”, guardam dados atómicos, assim como a versão sintetizada dos dados e são modelados num esquema em estrela.

Sem uma infraestrutura central que integre os data marts, esta abordagem de design depende de uma estrutura *bus* dimensional que garanta a integração lógica dos data marts. Esta integração é feita através do uso de dimensões e factos conformizados quando se constrói um novo data mart no sistema. Desta forma as dimensões e factos do cubo OLAP são reutilizadas de forma a otimizar a usabilidade e performance das queries, permitindo aos utilizadores a realização de uma única query a todos os data marts juntos.

O data warehouse torna-se uma realidade virtual em vez de uma realidade física, sendo que podem estar espalhados em diversos servidores e o data warehouse serve de entidade central, sendo a soma de todos os data marts do sistema.



As vantagens deste design são:



- Minimiza estruturas de dados redundantes para acelerar custos e implementação.
- Possui um bom balanço entre flexibilidade centralizada e localizada.
- Capacita a criação “user-friendly” de data marts com estruturas de dados flexíveis.
- As estruturas de dados partilhados juntamente com a a estrutura em *bus* eliminam o esforço repetido de criar múltiplos data marts numa estrutura não-arquitectada
- Uma vez que os data marts contêm dados atómicos, os utilizadores não necessitam de explorar de um data mart para um data warehouse para obter informação detalhada ou obter transacções.
- É facilmente extensível uma vez que usa um modelo de dados conformizado.

No entanto, este modelo não deixa de ter desvantagens na sua implementação, tais como:

- Obriga a que as organizações implementem forçosamente dimensões e factos standard em todos os seus departamentos para garantir uma integração dos dados correcta, havendo uma tendência natural para a criação de data marts independentes e não-integrantes em ambientes distribuídos ou descentralizados.
- Existem poucas ferramentas que conseguem agregar informação de vários data marts distintos fisicamente
- Considerando que este modelo foi concebido para otimizar as queries, não suporta *operational data stores*.

DESIGN HÍBRIDO

O design híbrido é uma tentativa de unificar o melhor das aproximações “top-down” e “bottom-up”. Tenta assim obter a velocidade e a orientação ao utilizador presente na aproximação “bottom-up”, sem sacrificar a integração imposta pelo warehouse na aproximação “top-down”.

Um data warehouse híbrido consiste em dois data warehouses interligados entre si, sendo um deles desenhado numa aproximação “top-down” e o outro desenhado numa aproximação “bottom-up”. Esta técnica tem como objectivo conseguir providenciar uma gestão de dados global para toda a empresa, assim como integração e exploração global dos dados. Ambos os warehouses armazenam dados na sua forma mais granular e atómica. Opcionalmente podem recorrer à mesma base de dados para os gerir.

O conceito base desta arquitectura é que todos os dados atómicos devem estar armazenados no data warehouse “bottom-up” no seu modelo normalizado de base de dados, sendo transferidos gradualmente para o data warehouse “top-down” à medida que são requisitados pelos gestores e analistas da organização.

O data warehouse normalizado capacita então um repositório de dados específico e ao mesmo tempo independente das suas fontes de dados, tornando-se útil para actividades de gestão de dados, tais como controlo de qualidade, auditoria, arquivação, etc. Serve também como um ponto de referência estável para o warehouse com modelo de dados dimensional, dado que o primeiro se mantenha actualizado regularmente, eliminando problemas técnicos relacionados com a captura de dados de variados sistemas de informação.

Poderemos considerar as seguintes vantagens desta abordagem::

- Providencia um desenvolvimento rápido dentro da framework arquitectural da empresa
- Desenvolve um modelo de dados empresarial através de várias iterações e apenas recorre a uma infraestrutura pesada quando realmente é necessária., tendo retorno de investimento mais rápido.
- Evita a criação de data marts não-integrados e independentes.
- Sincroniza meta-dados e modelos de base de dados entre as versões globais e locais dos dados.

Esta arquitectura, no entanto, apresenta alguns aspectos negativos:

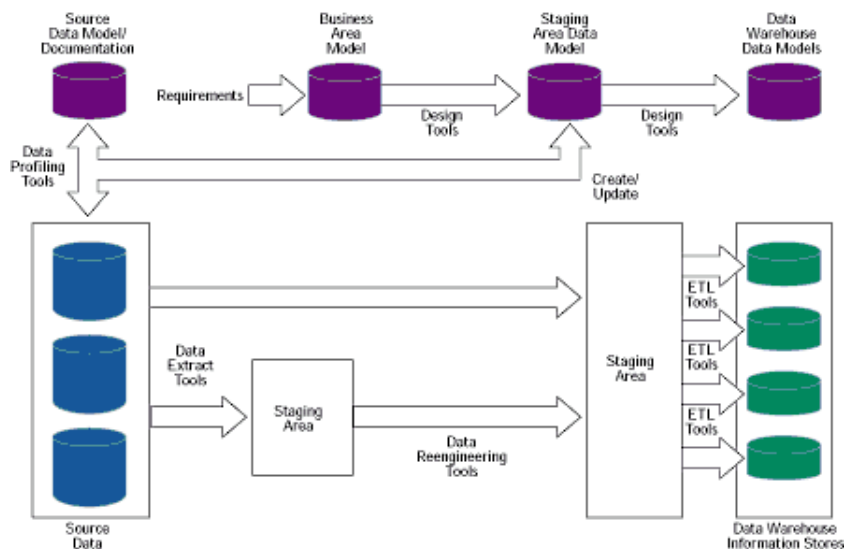


- Preencher um data warehouse dimensional através de outro warehouse normalizado pode ser altamente prejudicial, podendo nunca ser compensado em termos de recursos e tempo gastos.
- Poucas ferramentas conseguem realizar queries dinâmicas sobre dados atômicos e sobre dados sintetizados presentes em bases de dados diferentes.
- Depende demasiado de ferramentas ETL para sincronizar os meta-dados entre as versões locais e as versões globais empresariais dos mesmos, adicionando o dobro do processamento ETL para a manutenção do sistema.
- É difícil de medir o seu rendimento real devido à sobrecarga dos processos ETL e devido ao processo complexo de modelação de dados e do sistema no seu todo. Tipicamente é aconselhado a empresas federadas com requisitos de BI complexos.

DESIGN FEDERADO

Esta abordagem foi criada com o intuito de evitar os problemas que surgem a longo prazo nos sistemas de data warehousing baseados num design “bottom-up”, nomeadamente problemas de integração de dados à medida que os requisitos de negócio e complexidade dos sistemas vão evoluindo com o tempo.

Para atingir este objectivo, após a criação interactiva de data marts independentes (comum à abordagem “bottom-up”), é criado um data warehouse central que suporta um modelo de negócio comum que mapeia toda a informação armazenada e gerida pelo sistema. Este modelo de negócio comum reforça uma consistência nos nomes usados e nas definições de negócio usadas na amplitude do sistema e vai sendo actualizado à medida que novos data marts são integrados no sistema ou data marts existentes são actualizados nos seus modelos de dados. Exemplificamos seguidamente uma arquitectura federada de um data warehouse:



Outro uso deste modelo de negócio comum é em aplicações de BI e análise, uma vez que contém na sua estrutura meta-dados importantes sobre as regras de negócio e convenções usadas nos dados recolhidos. Estes meta-dados servem então de suporte para a camada de gestão e validação de dados de uma aplicação de BI, simplificando o desenvolvimento complexo deste tipo de software.



3.2 - PROCESSO DE ETL

Tal como já foi dito, o ETL é um processo relacionado com a criação e manutenção de data warehouses que envolve três fases:

- Extração (Extraction).
- Transformação (Transformation).
- Carregamento (Loading).

A primeira fase de extração corresponde à primeira fase do processo, onde são carregados dados de várias fontes de informação, com diferentes formatos e standards. Essas fontes tipicamente são bases de dados relacionais ou ficheiros de texto onde a informação é guardada em bruto. O objectivo desta fase passa por obter o conjunto de dados necessários e converte-los num único formato standard para proceder para a segunda fase do ETL: a transformação

Nesta segunda fase, são aplicados um conjunto de regras e funções sobre os dados extraídos, de forma a que no fim da transformação a agregação de dados recolhidos passe a obedecer a certos requisitos técnicos ou de negócio. Estes dados irão depois ser normalmente utilizados para efectuar processos de Business Intelligence. Dentro das transformações possíveis, podemos efectuar algumas das seguintes:

- Seleccionar apenas certas colunas dos dados recolhidos.
- Traduzir valores codificados (ex: 1 passa a ser True, 0 passa a ser False).
- Codificar valores (ex: masculino passa a ser "M").
- Aplicar fórmulas matemáticas para calcular novo valores.
- Filtragem de certo tipo de dados.
- Ordenação.
- Agregação de dados.
- Etc...

Finalmente na última fase os dados previamente transformados e agregados são carregados tipicamente num data warehouse. No entanto, dependendo da organização e das suas regras de negócio, alguns data warehouses podem sobrescrever valores caso haja conflitos de chaves ou valores iguais, enquanto que noutros warehouses a informação pode ser guardada de acordo com o seu historial de entrada na base de dados (visto que a informação é normalmente actualizada numa base diária, semanal ou mensal, conforme os requisitos da organização).

Em termos de oferta do mercado temos várias ferramentas de ETL e Data Washington, proprietárias e open source com funcionalidades bastante semelhantes no que toca a conectividade com bases de dados diferentes, transformações de dados, etc. No entanto existem vários factores relevantes que irão ajudar na escolha de uma dessas ferramentas para utilizar no nosso projecto:

- Performance
- Escalabilidade
- Estabilidade
- Interoperabilidade

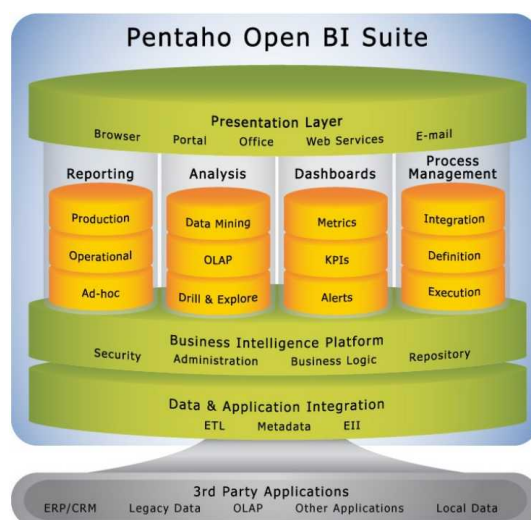


4 - PENTAHO BUSINESS INTELLIGENCE SUITE

A *Pentaho BI Suite* trata-se de um conjunto de produtos comerciais de software independentes, que no seu todo capacitam um espectro alargado de processos e funcionalidades que suportam variados requisitos de business intelligence.

A Pentaho BI Plataforma, componente nuclear desta suite, é orientada a processos, uma vez que o seu controlador central é um motor de workflow. Este motor é usado para definir processos de business intelligence, para mais tarde poderem ser executados dentro da plataforma, permitindo a customização e adição de novos processos conforme os nossos requisitos de negócio. A plataforma de BI também é orientada a solução, considerando que as operações da plataforma são especificadas na definição de processos de BI e em action documents que especificam todas as actividades que queremos executar.

Uma vez definidos os nossos processos, a *Pentaho BI Suite* dispõe de ferramentas para capacitar a análise interactiva de indicadores de negócio, através da criação de relatórios significativos e dashboards customizáveis. Destaca-se também a funcionalidade de tratamento e integração de dados de diversas fontes de informação, assim como funcionalidades de data mining, permitindo a escalabilidade e centralização de fontes dados de BI.



Esta suite consiste nos seguintes componentes:

- *Pentaho BI Server* – Este servidor é uma plataforma de BI de classe empresarial que suporta todas as funcionalidades disponíveis para o utilizador final, tais como a criação de relatórios e dashboards e ainda análise detalhada, juntamente com funcionalidades de segurança, integração, agendamento e workflow presentes no back-end
- *Pentaho Administration Console* – É uma aplicação standalone distribuída na plataforma de BI que providencia aos administradores do sistema uma interface leve para administrar utilizadores, papéis, fontes de dados, agendamentos no servidor, etc.
- *Pentaho Design Studio* (ferramenta de cliente) – É uma colecção de editores e visualizadores integrados numa única aplicação que disponibiliza um ambiente gráfico para construir e testar Action Sequence Documents.
- *Pentaho Metadata Editor* (ferramenta de cliente) – É uma ferramenta que constrói domínios e modelos de meta-dados para a plataforma da Pentaho. Um modelo de meta-dados mapeia a estrutura física da base de dados para um modelo lógico de negócio.



- *Pentaho Report Designer* (ferramenta de cliente) – É a ferramenta principal para criar e publicar relatórios Pentaho. Providencia uma interface gráfica que permite ao utilizadores interligar os seus relatórios de dados, de desenho e de pré-visualização
- *Pentaho Schema Workbench* (ferramenta de cliente) – É a ferramenta principal para desenhar, editar e publicar schemas OLAP para o Pentaho Analysis(Mondrian).
- *Pentaho Aggregation Designer* (ferramenta de cliente) – É o ambiente gráfico usado para aumentar a performance das queries num schema OLAP do Mondrian, através da criação de tabelas de agregação.
- *Pentaho Analyser* (ferramenta de cliente) – É uma ferramenta interactiva de análise (ou visualizador OLAP) que permite utilizadores orientados ao negócio a criação de relatórios baseados na web de forma interactiva, atractiva e significativa.
- *Pentaho Data Integration* (ferramenta de cliente) – É o ambiente gráfico drag-and-drop de desenho que potencializa as capacidades de extracção, transformação e carregamento (ETL) recorrendo a uma abordagem orientada a meta-dados.



4.1 - PENTAHO SERVER

O *Pentaho Server* tem como requisitos de software executar dentro de um servidor web compatível com J2EE, executando no caso da *Pentaho BI Suite* dentro do *Jboss AS* já previamente configurado pela instalação.

É este servidor que permite as variadas funções da plataforma de BI, para poderem ser apresentadas aos clientes de forma consistente e user-friendly. O conteúdo das suas diferentes componentes pode ser obtido como XML, HTML, ou apresentadas através das JSR-168 portlets presentes no java. Estas portlets podem ser embebidas em qualquer portal que suporte os standards JSR-168 tais como o *IBM WebSphere*, *OracleAS Portal* ou *BEA WebLogic Portal*. As stylesheets XSL e CSS usadas por esses componentes são também acessíveis e totalmente customizáveis.

Os variados componentes de BI que constituem servidor servem para, no seu conjunto, resolver um problema de BI criando uma solução. Dentro da solução obtida, o comportamento, interoperabilidade e interação do utilizador em cada sub-sistema é definido por uma colecção de *Solution Definition Documents*.

Estes ficheiros são documentos XML que contêm:

- Definições de processos de negócio (no standard XPD)
- Definições de actividades que executam arbitrariamente como parte dos processos ou chamadas por webservices. Estas actividades incluem definições para fontes de dados, queries, templates de relatórios, regras de entrega e notificação, regras de negócio, dashboards e visões analíticas.
- As relações entre todos os elementos descritos acima.

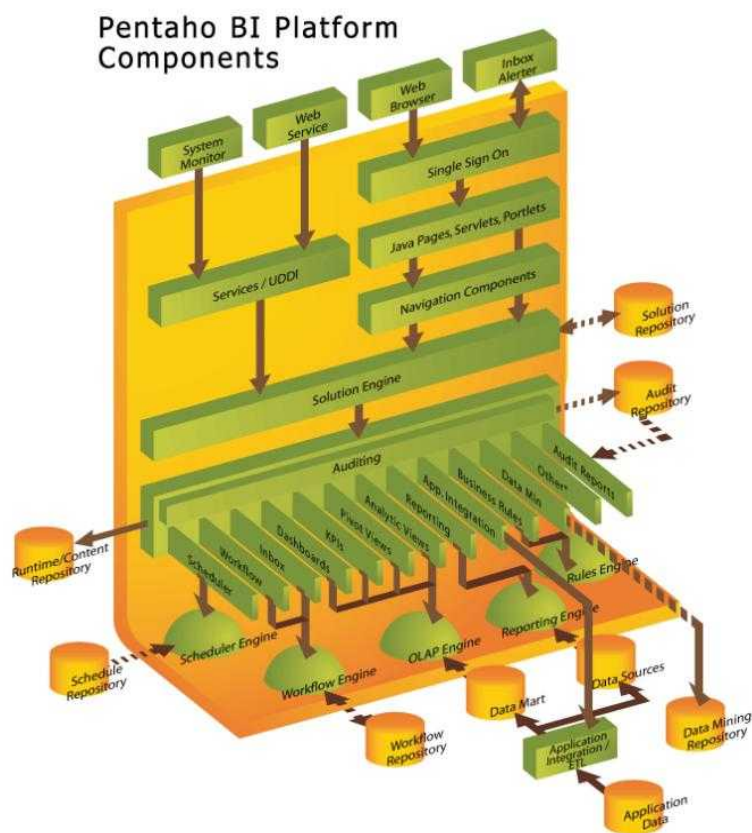
Os componentes presentes no servidor dependem de um motor de soluções para obter informação sobre os documentos de soluções disponíveis, para garantir segurança, para obter informação sobre relatórios e itens de workflow, assim como para obter informação sobre dados e auditorias. Mais do que uma solução pode executar no servidor.

O servidor contém também uma infra-estrutura que providencia um mecanismo de administração de sistema avançado. Dentro deste mecanismo estão incluídos serviços de monitorização de sistema (SNMP), relatórios de utilização, suporte para web services, ferramentas de validação de configuração e ainda ferramentas de diagnóstico.

Estão incluídos também no servidor sistemas e componentes que providenciam suporte para análise e criação de relatórios sobre a performance de processos BI. Inclui também técnicas de “slice-and-dice” e “what-if”, assim como de data mining, que podem ser aplicadas sobre atributos de itens de workflow, tarefas individuais, empregados e serviços envolvidos nas tarefas de workflow.

O servidor também suporta Enterprise Application Integration (EAI) para integração em tempo real com aplicações operacionais, assim como capacidades de Extract, Transform and Load (ETL) para criação de data warehouses e data marts.

Seguidamente mostramos um esquemático da arquitectura da plataforma BI.



Como podemos observar, o motor de soluções é o centro de toda a arquitectura e é a partir desta camada que é gerido o acesso aos diferentes componentes BI. Os diversos serviços disponíveis providenciam web services a aplicações externas e têm acesso ao mesmo motor de soluções que os componentes da interface de utilizador. Estes serviços são lançados pelo motor de workflow e pelo agendador de tarefas para executar acções dentro do sistema.

Todas as auditorias executadas são directamente construídas nos componentes da plataforma. A plataforma consegue providenciar relatórios de performance de processos ao extrair informação histórica ou em tempo real dos repositórios de auditoria e de workflow.

Cada motor presente na arquitectura tem um componente a que lhe corresponde e que é responsável pela integração do motor na plataforma. Os motores em si podem ser trocados ou até mesmo adicionados à plataforma se os componentes necessários são criados.

A Pentaho BI Platform integra-se com sistemas externos que disponibilizam dados ao motor responsável pelos relatórios e recebe eventos do motor de workflow. A plataforma também disponibiliza monitorização do sistema através do Simple Network Management Protocol (SNMP).

Os repositórios são armazenados dentro de uma RDBMS exterior à plataforma. Os repositórios embebidos são guardados em bases de dados open-source, nomeadamente FireBird e MySQL, podendo ser substituídas por outras conforme os requisitos impostos.

Variados motores de regras são disponibilizadas na Pentaho BI Platform para que a lógica de negócio esteja exposta e seja facilmente customizável. Outros motores de regras podem ser adicionados ao sistema. Os motores de regras de negócio são externos aos componentes presentes e podem ser usados por qualquer um desses componentes, sem restrições.



Dentro da plataforma de BI podemos encontrar o componente responsável pelos processos de ETL, assim como de Data Warehousing, o *Pentaho Data Integration*, no qual nos iremos focar mais tarde.

Nem todos os componentes estão incluídos no diagrama, outros componentes incluem e-mail, impressão, formatação de mensagens, gestão de atributos de instâncias de workflow, reportagem de performance de processos e análise “what-if”.

Algumas partes da arquitectura usam uma combinação de tecnologias que podem ser sempre trocadas por tecnologias equivalentes:

- O servidor J2EE usado é o Jboss AS.
- As interfaces da plataforma de BI são feitas através de Java Server Pages (JSPs), servlets e portlets.
- A BI Suite inclui um motor OLAP open source, o Mondrian, que pode ser trocado por qualquer servidor OLAP complacente com MDX.
- A plataforma também possui motores de regras baseados em javascript e SQL

Sendo assim, a Pentaho BI Platform é uma conjunção de diversos componentes open source tais como:

- Mondrian OLAP Server – Um servidor de base de dados dimensional.
- jPivot Analysis Front-End – Uma biblioteca JSP que permite fazer navegações típicas de bases de dados OLAP, tais como slice-and-dice, drill-down e roll-up, usando o Mondrian como servidor.
- Firebird RDBMS – A base de dados utilizada para todos os repositórios internos do servidor.
- Enhydra JaWE – Um editor gráfico em java-swing de workflow WfMC e XPDL.
- Enhydra Shark – O servidor de workflow que suporta as criações do Enhydra JaWE.
- Kettle EII and ETL (ou *Pentaho Data Integration*) – O componente de ETL e de data warehouse que será abrangido mais tarde no relatório.
- Jboss Application Server, Jboss Hibernate e Jboss Portal – Os componentes base do servidor Pentaho.
- Weka Data Mining – Um conjunto de algoritmos inteligentes de data mining.
- Eclipse Workbench e componentes BIRT – Um sistema de criação de relatórios (reporting) baseado no eclipse e especializado para aplicações web.
- JOSSO single sign-on – Componente especializado na autenticação single sign-on de utilizadores na suite e de providenciar suporte para autenticações LDAP (Lightweight Directory Access Protocol)
- Mozilla Rhino Javascript Processor – Um interpretador de javascript internamente a partir da java virtual machine.
- Quartz – Um serviço de agendamento de tarefas.

A plataforma também suporta um conjunto de protocolos e standards abertos, incluindo XML, JSR-94, JSR-168, SVG, XPDL, XForms, MDX, WSBPEL, WSDL e SOAP.

Em relação aos repositórios embebidos no servidor, temos três repositórios que são responsáveis pela definição, execução e auditoria de uma solução respectivamente:

- Repositório de Soluções – Onde são guardados os meta-dados que definem as soluções.
- Repositório de Runtime – Onde são guardados itens de trabalho que o motor de workflow gere.
- Repositório de Auditorias – Onde é guardada e monitorizada a informação relativa a auditorias.

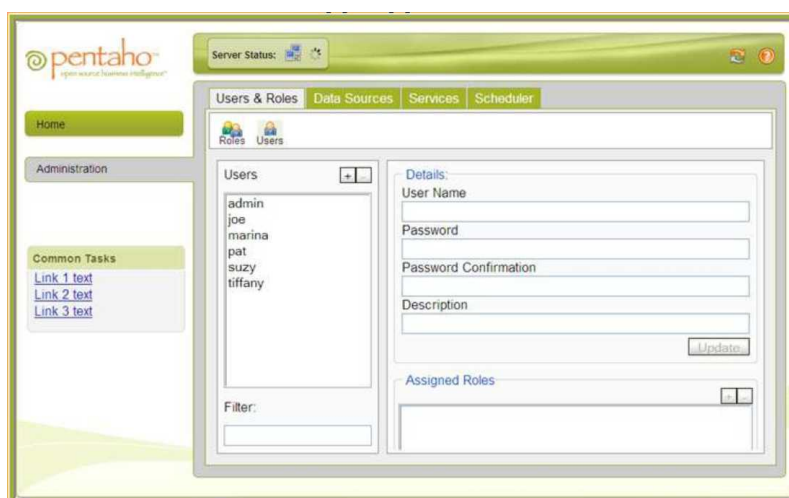


4.2 - PENTAHO ADMINISTRATION CONSOLE

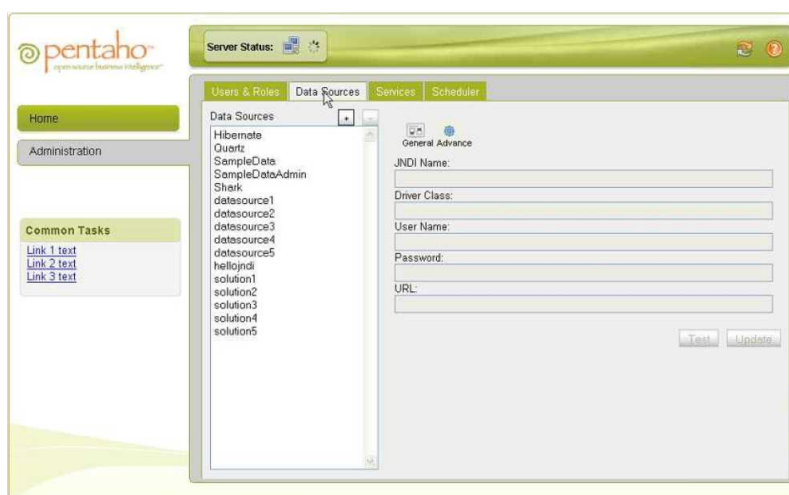
Esta consola é um servidor que serve como front-end gráfico para tarefas administrativas sobre o *Pentaho Server*, podendo correr numa máquina diferente à do servidor *Pentaho* e facilitando a gestão de segurança da suite.

Está baseado no *Jetty*, um webserver e contentor de Java servlets, semelhante ao *Tomcat*, embora seja mais leve na sua implementação, tornando-o especializado em integrações em software já existente.

Permite a gestão de utilizadores e grupos de utilizadores, assim como a gestão dos seus papéis e permissões dentro do servidor.



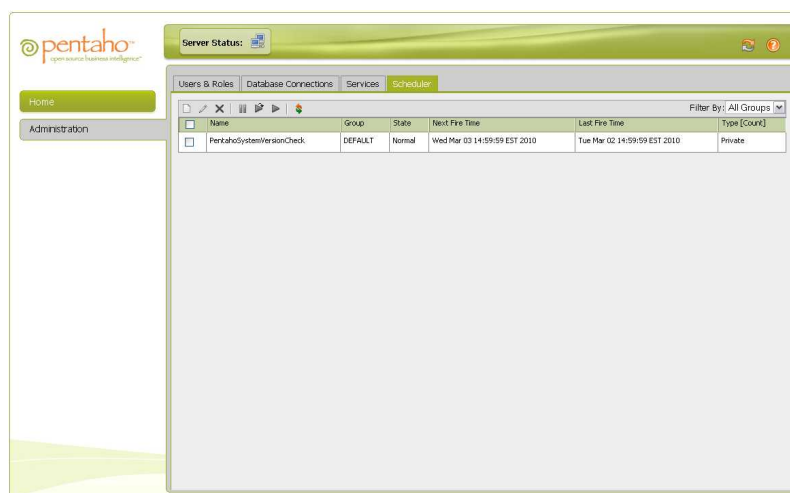
Também permite a gestão das fontes de dados a que o servidor pode aceder, podendo configurar variadas fontes JDBC para funcionar em conjunto com o servidor.





A principal funcionalidade desta consola passa pelo agendamento de tarefas a efectuar no servidor, gerido pelo motor open source *Quartz*. É possível gerir as tarefas a decorrer no servidor, monitorizando, suspendendo, resumindo, criando e apagando agendamentos conforme as nossas necessidades.

Existe a possibilidade de agendar tarefas mais detalhadamente através de action sequences criadas através do *Design Studio*, providenciando um maior controle sobre o fluxo de tarefas a decorrer. Estas uma vez criadas podem ser atribuídas a um agendamento criado pelo administrador, desde que esse utilizador tenha as permissões para tal sendo designada de uma *subscrição*.



Outras capacidades desta consola incluem ferramentas que permitem limpar a cache de dados e schemas do *Mondrian* usadas no servidor, refrescar definições de sistema globais, refrescar meta-dados usados nos relatórios, visualização de relatórios, etc.





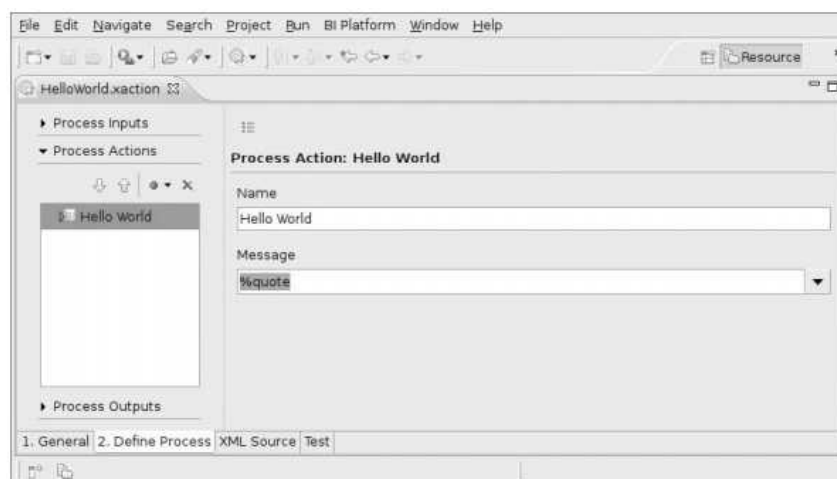
4.3 - PENTAHO DESIGN STUDIO

É uma aplicação java de desktop baseada no eclipse, orientada a workspaces e projectos, com um plugin especializado na criação e manutenção de *action sequences*.

Uma *action sequence* é um conjunto de acções pré-determinadas que podem ser executadas do *Pentaho BI Server*. A execução de uma *action sequence* pode ser activada por uma acção do utilizador, um agendador, ou qualquer outro evento, incluindo outra *action sequence*. Estas podem variar de complexidade, podendo ser simples, por exemplo, “executar um relatório” ou “activar o aparecimento de uma mensagem no écran”, como também poderia ser complexa, como por exemplo “encontrar todos os clientes com dívidas e enviar-lhes um memorando num formato específico de ficheiro, contendo detalhes sobre a dívida”.

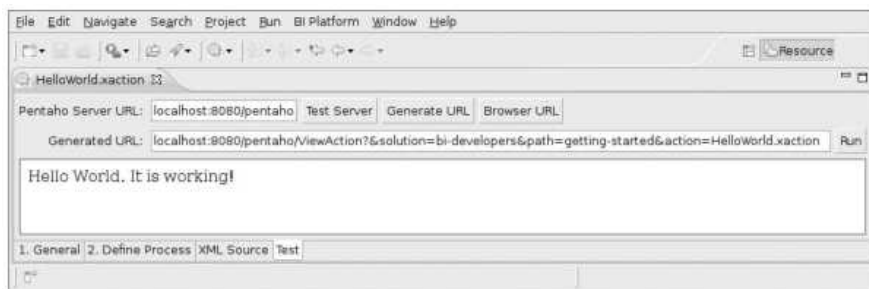
Também podem definir acções de baixo-nível, como por exemplo, definir variáveis de ambiente quando um utilizador faz login, ou criar uma lista geral de parâmetros para serem usados por outro processo ou *action sequence*.

Uma vez criado o nosso workspace e projecto, ao criarmos uma *action sequence* deparamo-nos com uma interface semelhante à imagem seguinte (nesta caso é uma *action sequence* apenas para mostrar uma mensagem “hello world”):



Na janela de edição de *action sequences* deparamo-nos com uma janela com várias perspectivas:

- Perspectiva geral, podendo alterar detalhes gerais, tais como nome da acção, versão, autor, tipo de resultado da acção, etc.
- Perspectiva de definição de processo, onde editamos a *action sequence* em si, estando divididas em três janelas:
 - Definição de inputs e recursos para os componentes que usarmos no nosso processo.
 - Acções do processo, onde detalhamos a sequência que queremos construir, disponibilizando uma grande variedade de componentes para construir a nossa acção, assim como um écran lateral para editar os detalhes de cada acção.
 - Definição de outputs da *action sequence*.
- Perspectiva de XML, onde podemos ver o XML gerado automaticamente da nossa acção.
- Perspectiva de teste, onde é permitido testar as nossas acções no servidor BI.



Algumas funcionalidades de componentes disponíveis no desenho de *action sequences* são:

- Obtenção de dados de bases de dados relacionais, OLAP, XML, MQL, Javascript e até mesmo do componente *Pentaho Data Integration*.
- Criação e edição de variados relatórios e gráficos produzidos na suite do Pentaho.
- Envio para e-mail e impressora.
- Ações relacionadas com o agendamento de tarefas (monitorização, criação, pausa, resumo de tarefas individuais do servidor, ou até do próprio agendador de tarefas).
- Execução de comandos SQL, processos BI do Pentaho, tarefas do *Pentaho Data Integration*, comparação de sets de resultados, etc.
- Mecânicas de controlo de fluxo, como *if statements* e *loops* que podem ser usados entre todas as ações indicadas acima.



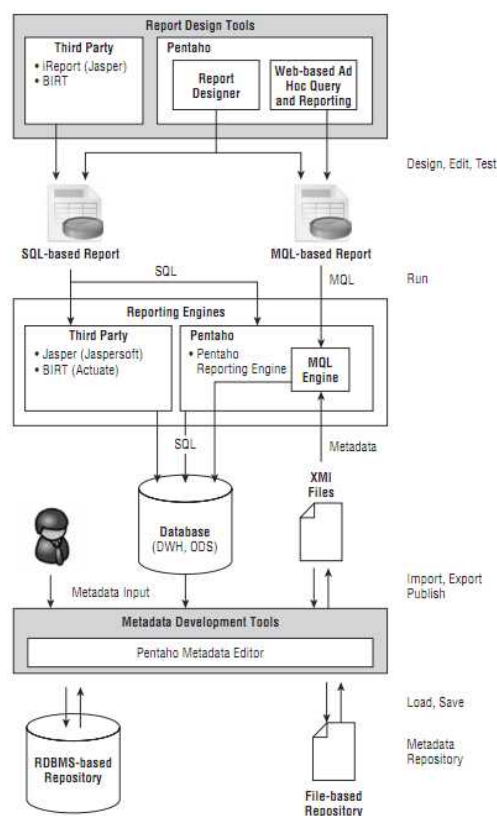
4.4 - PENTAHO METADATA EDITOR

O *Pentaho*, de forma a reforçar o uso de um formato de dados e comportamento consistentes, recorre a uma camada de meta-dados. Esta camada é que permite a grupos de propriedades comportamentais e visuais, denominadas de *conceitos*, interligarem-se a objectos de dados tais como tabelas e colunas, impondo regras sobre os dados antes de serem tratados pela camada de negócio.

Todos os inputs de meta-dados provenientes de bases de dados, assim como meta-dados definidos pelo utilizador são criados através do *Pentaho Metadata Editor* e guardados num repositório. Actualmente este repositório está separado do repositório responsável pelos processos de integração de dados, sendo dedicado apenas a actividades de criação de relatórios.

Meta-dados também podem ser exportados do repositório e armazenados sob a forma de ficheiros .xmi, ou numa base de dados. Estes estão associados com uma determinada solução num servidor *Pentaho*, onde podem ser usados como um recurso para serviços de reportagem baseados em meta-dados, como forma de abstracção dos detalhes físicos dos dados ou estruturas físicas de armazenamento sobre os quais actuam.

Mostramos então esquematicamente o contexto deste editor dentro da plataforma:



Este editor organiza a camada de meta-dados em domínios de meta-dados. Cada domínio é uma colecção de objectos de meta-dados que podem ser usados numa solução dentro do *Pentaho*.

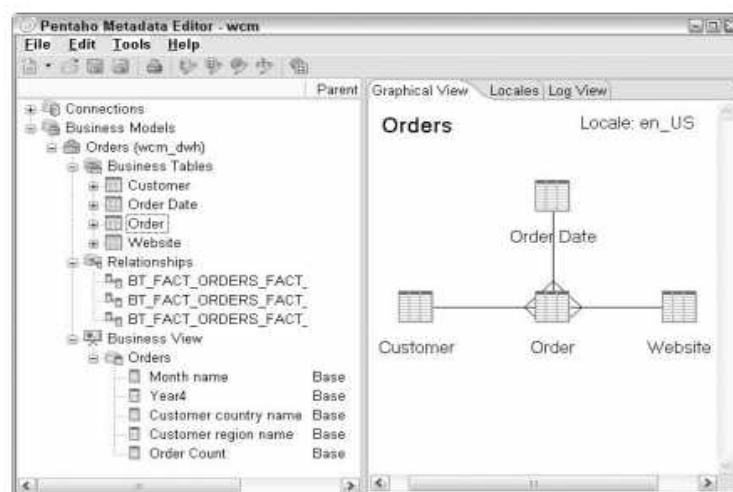


Este editor permite então definir vários objectos e suas propriedades duma maneira transparente:

- Descrições de conexões a variadas bases de dados, assim como as suas linhas e colunas das tabelas.
- Modelos de negócio, focados em diferentes áreas específicas de negócio, de forma similar a um data mart, para que mais tarde possam ser usadas em relatórios pelos utilizadores.

Dentro dos modelos de negócio podemos definir:

- Tabelas e colunas de negócio, derivadas das tabelas e colunas físicas das bases de dados definidas anteriormente.
- Relações que definem a junção entre duas tabelas de negócio, permitindo a interligação lógica das tabelas do nosso modelo.
- Perspectivas de negócio, dentro das quais definimos várias categorias de negócio. Estas categorias representam um conjunto coerente de colunas de negócio que estão relacionadas entre si de alguma maneira, para mais tarde serem usadas na criação de relatórios (duma perspectiva baixo-nível, podemos dizer que representa uma conjunção de dimensões relevantes de um cubo OLAP).



Uma vez criado o nosso domínio de meta-dados, podemos exportá-lo sob a forma de um ficheiro XML para mais tarde colocarmos na pasta respectiva da solução *Pentaho* (podendo apenas ser usado um ficheiro por solução) para a qual modelámos este domínio, ou podemos carregá-la directamente para o servidor através da interface do próprio editor de meta-dados. Estes ficheiros também podem ser importados para futuras edições sobre o editor.



4.5 - PENTAHO REPORT DESIGNER

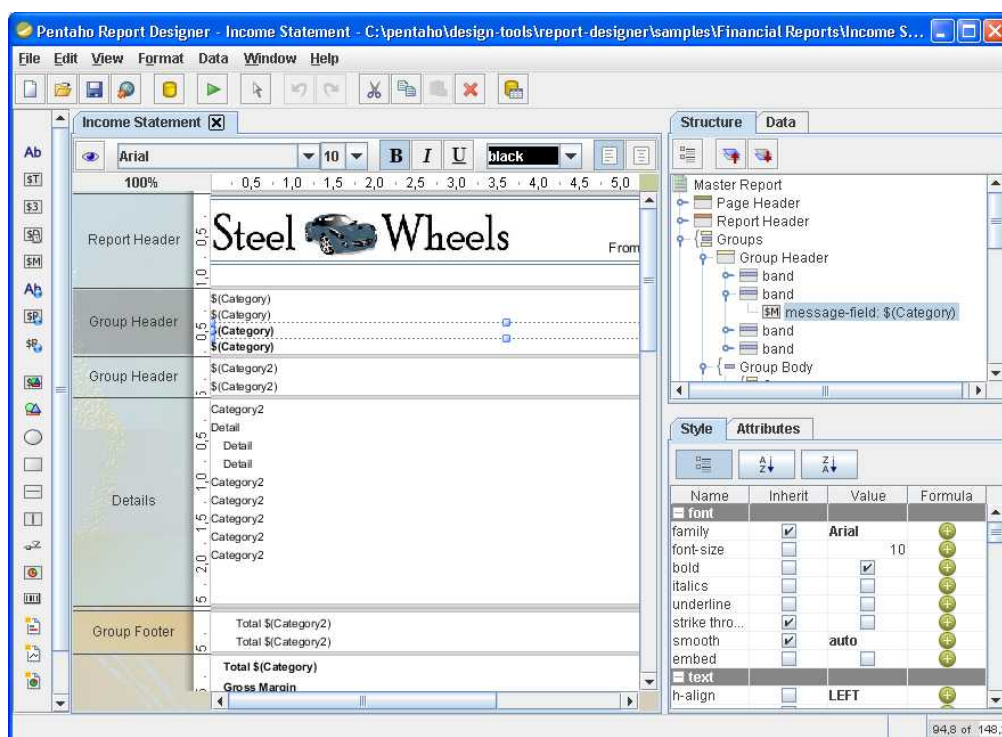
A suite da *Pentaho* foi construída de maneira a providenciar capacidades de *agile reporting* através da sua interface web. Esta maneira de produzir relatórios ad-hoc limita bastante o utilizador a nível de estruturação e customização dos relatórios devido às interfaces web talhadas para simplificar o uso dos diferentes componentes da plataforma *Pentaho*.

No entanto, existe a possibilidade de detalhar e customizar os relatórios criados através do ambiente gráfico do *Pentaho Report Designer*. Esta ferramenta distingue-se pela sua capacidade de gerar relatórios usando modelos de meta-dados previamente criados no *Pentaho Metadata Editor* como fontes de informação. Estes relatórios também podem ser directamente publicados através desta aplicação para poderem ser usados no *Pentaho User Console* (também conhecido por *Pentaho Analyser*), front-end gráfico do *Pentaho Server*.

Apresenta-se como um editor orientado a “listras”, ou seja, divide os relatórios em vários agrupamentos de dados onde podem ser colocados diferentes elementos do relatório, semelhante ao software *Crystal Reports*. Após desenharmos o nosso relatório duma perspectiva de desenho da sua estrutura, podemos fazer uma pré-visualização do resultado final em PDF, HTML, XLS, RTF e CSV.

Ao lançarmos a aplicação deparamo-nos com um écran inicial onde somos questionados se queremos um relatório em branco ou criar um através de um wizard. Este wizard permite-nos criar passo a passo um relatório, permitindo inicialmente escolher um template numa lista ou adoptar o design de um ficheiro de relatório já existente. Após esta escolha, seleccionamos uma base de dados da qual queremos retirar os dados que pretendemos, para depois os podermos agrupar e colocar no relatório.

Em termos de elementos de desenho, situados na barra direita da aplicação, são elementos comuns a qualquer aplicação de criação de relatórios deste tipo. Possui também um campo de estrutura geral do relatório, situado no canto superior esquerdo da aplicação, onde é representada a hierarquia de estrutura do mesmo, assim como um separador referente a dados. Neste separador é que controlamos os dados importados da base de dados ou meta-dados, assim como podemos usar um conjunto de funções sobre os dados antes de serem mostrados no relatório em si.





Os relatórios são divididos em várias secções, onde são controlados diferentes aspectos do relatório:

- *Page Header/Footer* – Contém informação que irá estar presente em todas as páginas do relatório, tipicamente números de página, datas, logos da empresa, etc.
- *Report Header/Footer* – Contém informação que apenas irá aparecer apenas uma vez. É normalmente usado para colocar resumos do relatório, parâmetros globais ou título.
- *Group Header/Footer* – Um relatório tem de ter pelo menos um grupo para organizar o seu conteúdo. Cada grupo contém um cabeçalho e um rodapé para mostrar etiquetas e sub-totais Grupos podem ser incorporados noutros grupos, criando uma hierarquia de estrutura no relatório.
- *Details Body* – Um dos grupos mais internos contém um destes campos onde as linhas individuais do resultado de uma query podem ser colocados. Também contém um cabeçalho e rodapé.

A criação de sub-relatórios também é permitida, sendo um dos elementos distintos de desenho desta aplicação. Este elemento foi criado de forma a contornar a limitação desta ferramenta, tendo em conta que editores deste género que recorrem ao agrupamento de dados em secções costumam ser limitados à partida na personalização do relatório, uma vez que limitam os dados a grupos e sub-grupos.

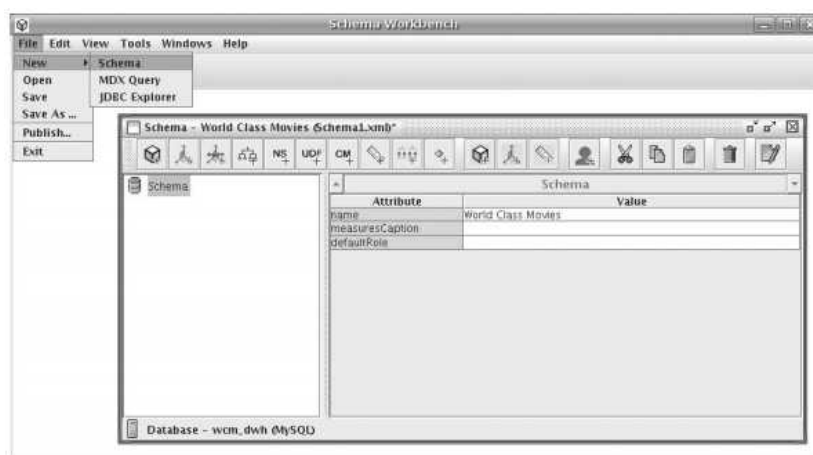


4.6 - PENTHAO SCHEMA WORKBENCH

Este software consiste numa aplicação gráfica simples para modelação de schemas para o servidor *Mondrian*, definidos em ficheiros XML.

Após definir a nossa ligação à base de dados, podemos explorá-la através do *JDBC Explorer*, mostrando a base de dados numa estrutura em árvore.

Ao seleccionarmos a opção de criar um novo schema, somos deparados com uma janela de edição gráfica com vários elementos de desenho na sua barra superior como mostra a figura:



Podemos seleccionar qualquer elemento para aparecer um menu de contexto na própria janela que permite a definição de parâmetros específicos ao componente. Também é possível mudar para modo de visualização de texto, onde podemos ver o código XML à medida que este é gerado pela ferramenta.

Este criador de schemas OLAP permite:

- Criação de cubos.
 - Escolha de tabelas de factos.
 - Adição de medidas.
- Criação de dimensões normais ou partilhadas
 - Edição da hierarquia por omissão e escolha de tabelas de dimensão.
 - Definição de níveis de hierarquia.
 - Opcionalmente adicionar mais dimensões.
- Adicionar dimensões a cubos.

A ferramenta também dispõe de um componente para efectuar queries MDX de forma a testar a funcionalidade dos schemas criados através da ferramenta.

No final de alguns testes elementares, possibilita também a publicação do schema para o servidor BI do *Pentaho* directamente.



4.7 - PENTAHO AGGREGATION DESIGNER

Esta aplicação standalone serve para tonificar a performance do servidor *Mondrian* através da criação de tabelas agregadas. Com bases de dados grandes, apesar das funcionalidades de cache do *Mondrian*, demora uma quantidade de tempo considerável ao navegar dentro de um cubo, uma vez que é necessário percorrer milhões de linhas nas tabelas, de cada vez que um novo nível de dimensão é seleccionado

dim_date (3,660)	dim_dvd (149,710)	dim_customer (145,371)	
date_key	dvd_key	cust_key	Revenue
20090723	12345	1123	5
20090830	3456	4432	5
20090915	11234	5543	7
20090924	44332	4456	3.50
20091012	33434	6654	5
20090821	43554	3222	3.50
20090514	55667	4455	5
1,359,267 fact rows			

Year	Quarter	Month	Genre	Country	Sum (revenue)
2009	Q3	Nov	SciFi	US	
10	40	120	38	2	
120 · 38 · 2 = 9,120 aggregate rows (-99.32%)					

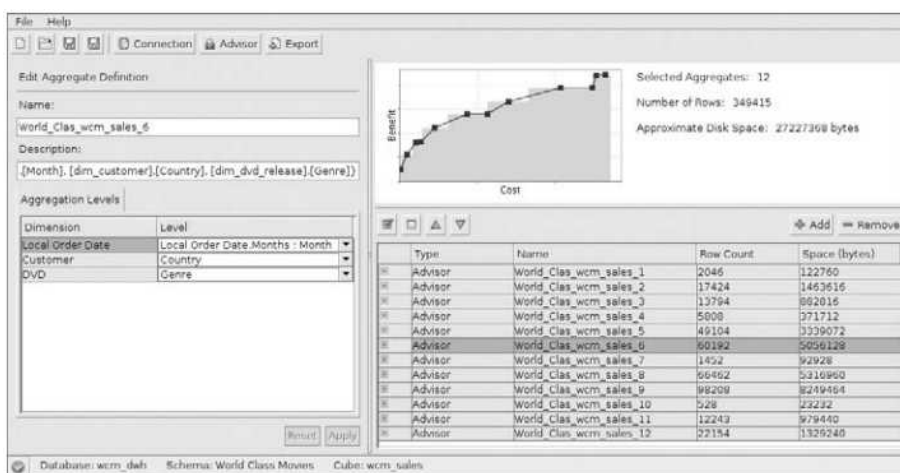
Year	Genre	Country	Sum (revenue)
2009	SciFi	US	
10	38	2	
10 · 38 · 2 = 720 aggregate rows (-99.95%)			

As tabelas agregadas são então usadas para pré-agregar dados para o *Mondrian* para determinadas dimensões.

Neste exemplo, numa situação normal teríamos que percorrer 1,359,267 linhas na base de dados para navegar numa tabela de factos com dados relativos a encomendas. No entanto, se agregarmos previamente as linhas de factos detalhadas a nível, por exemplo, de mês/género/país, teríamos apenas de percorrer 9,120 linhas para explorar todas as combinações de dados nesse nível.

Esta criação e actualização de tabelas agregadas pode ser mantida em background através da criação de uma tarefa no *Pentaho Data Integration* com execuções periódicas de forma a manter os dados consistentes.

Esta ferramenta foca-se na execução de um algoritmo inteligente que sugere a criação de determinadas tabelas agregadas, conforme as tabelas que decidimos otimizar, contornando a sua criação manual através do *Pentaho Schema Workbench*.



Podemos então:

- Criar e editar tabelas agregadas manualmente e subsequentemente seleccionar níveis de agregação para cada uma delas.
- Pré-visualizar, executar e exportar scripts de criação / carregamento das tabelas.
- Publicar o schema otimizado do *Mondrian* para o servidor *Pentaho*, ou exportá-lo para um ficheiro.



4.8 - PENTAHO ANALYSER

Também conhecido pelo *Pentaho User Console*, é uma aplicação web que serve de front-end para os utilizadores orientados ao negócio efectuarem análises sobre os cubos OLAP presentes num data warehouse.

Após fazermos login com o nosso utilizador na página inicial temos a opção de criar relatórios standard, relatórios analíticos e dashboards.

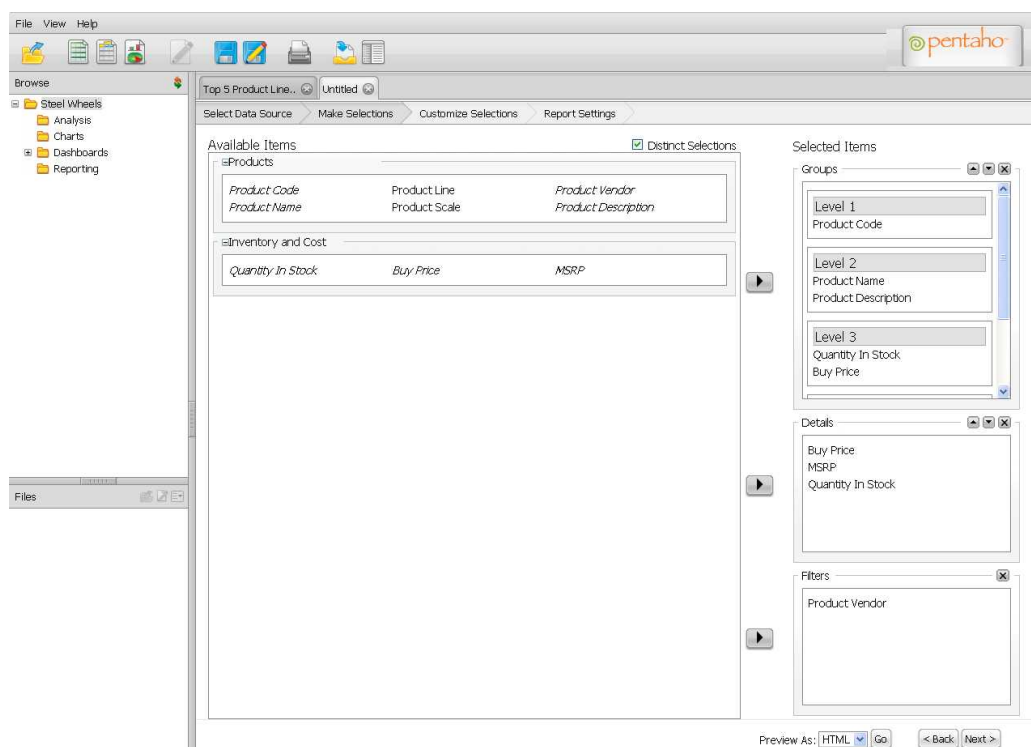
Outra opções interessantes nesta interface são:

- A possibilidade de agendar a criação de relatórios, dashboards, etc. através de action sequences criadas no *Design Studio*.
- Partilha de dashboards e relatórios entre vários utilizadores.
- Monitorização de execuções em background do servidor.

criação de relatórios standard

Na criação de relatórios standard, somos guiados através de um wizard para seleccionar inicialmente um template para usarmos, seguido da nossa fonte de dados. Estas são representadas numa perspectiva de negócio, apenas ditando o nome do cubo/modelo e os dados que contêm numa breve descrição textual. Neste passo também se pode escolher qual o formato da pré-visualização do relatório (HTML, PDF, Excel ou CSV).

Seguidamente somos apresentados com uma interface drag-and-drop onde arrastamos os vários elementos do cubo para vários níveis de agrupamentos de dados (podendo cada uma representar uma linha ou coluna na tabelas de informação geradas). Podemos depois escolher na caixa de detalhes os dados que queremos pormenorizar em bruto. Por fim, seleccionamos alguns elementos para a caixa de filtros, para mais tarde podermos escolher filtros para gerar relatórios ad-hoc de um subset dos dados dos relatório.





No passo seguinte podemos alterar opções de formatação individualmente para cada um dos elementos usados do cubo, assim com activar o uso de funções, uso de restrições e formatações prévias sobre os dados, alterar a ordenação dos dados das tabelas geradas pelos campos detalhados que escolhemos no passo anterior, etc.

Por fim, no último passo antes da publicação podemos alterar alguns detalhes gerais do relatório, tais como a orientação de página, tipo de papel, definição de cabeçalhos e rodapés, etc.

Após a definição destas opções de formatação de página, obtemos o nosso relatório, juntamente com os filtros que disponibilizámos anteriormente, assim como a opção de definir o tipo de output do relatório:

The screenshot shows the Pentaho Report Designer application. The left sidebar contains a 'Browse' tree with folders for 'Steel Wheels', 'Analysis', 'Charts', 'Dashboards', 'Widget Library', 'Analysis Views', 'KPIs', 'Report Snippets', and 'Reporting'. The main workspace displays an 'Invoice Statement' report. The 'Report Parameters' section at the top includes 'Select Customer' (Australian Gift Network, Co), 'Report Stamp' (Review), and 'Output Type' (PDF). Below this, there are 'View Report' and 'Auto-Submit' checkboxes. The report content includes the 'Steel Wheels' logo and contact information, a 'TO:' section with the customer's address, an 'INVOICE' header, and a table of items ordered. A 'Payment History' table is also visible at the bottom left, and a yellow box on the bottom right prompts the user to 'Send Payment and Remittance Slip to:'.

Steel Wheels
500 International Speedway, Daytona Beach FL 32114
(123) 456-7890 http://www.steelwheels.com

TO: Australian Gift Network, Co
31 Duncan St. West End,
South Brisbane, Queensland 4101 Australia

INVOICE

Attn: Tony Calaghan Invoice #: 10152
Sales Rep: 1611 Account Number: 333
Terms: Net 30 days Date: Setembro 25, 2003

SKU	Product Description	Price/Unit	Qty Ordered	Total Price
S18_4027	1970 Triumph Spitfire		35	\$4,524.10
S32_3207	1950's Chicago Surface Lines Streetcar		33	\$1,681.35
S24_4048	1992 Porsche Cayenne Turbo Silver		23	\$2,802.09
S24_1444	1970 Dodge Coronet		25	\$1,632.75
				\$10,640.29

Payment History

Date	Check#	Amount
11-15-03	HL209210	\$ 27,099.00
10-17-03	JK479662	\$ 10,640.00

Send Payment and Remittance Slip to:
Steel Wheels
500 International Speedway
Daytona Beach, FL 32114



A criação de relatórios analíticos tem um propósito bastante diferente dos relatórios standard.

Enquanto que os relatórios standard tentam de certa forma, replicar de forma simplificada numa interface web a habilidade de desenho de relatórios completos do *Pentaho Report Designer*, os relatórios analíticos permitem o desenho ad-hoc directo de tabelas numa interface drag-and-drop, para tornar mais directa actividades de análise.

Após escolhermos o schema da base de dados pretendido e o seu respectivo cubo, somos levados para uma janela de desenho, onde encontramos do lado esquerdo os elementos representativos das várias dimensões e medidas do cubo e do lado direito a área de desenho propriamente dita.

Ao arrastarmos os elementos, podemos criar novas colunas ou filas da tabela com esses dados, consoante as nossas necessidades, podendo agregar vários níveis de colunas ou filas de forma a hierarquizar os dados em diferentes perspetivas.

Também é possível, para cada coluna gerada ou medidas usada, aplicar diferentes opções de ordenação de dados, assim como filtros de forma a limitar o espectro dos dados visíveis e a opção de visualizarmos os dados em gráficos.

The screenshot displays the Pentaho Report Designer software interface. On the left is a 'Browse' pane showing a folder structure with 'Steel Wheels' selected. The main workspace shows a report titled 'Top 5 Product Lines by Territory' with a table of sales data. The table has columns for Territory, Line, and sales/quantity data for the years 2003 and 2004. The data is grouped by territory (APAC, EMEA, Japan, NA) and then by product line (Classic Cars, Vintage Cars, Motorcycles, Trucks and Buses, Planes). The right pane shows 'Available fields (14) for: SteelWheelsSales', including dimensions like Customer, Markets, and Time, and measures like Sales.

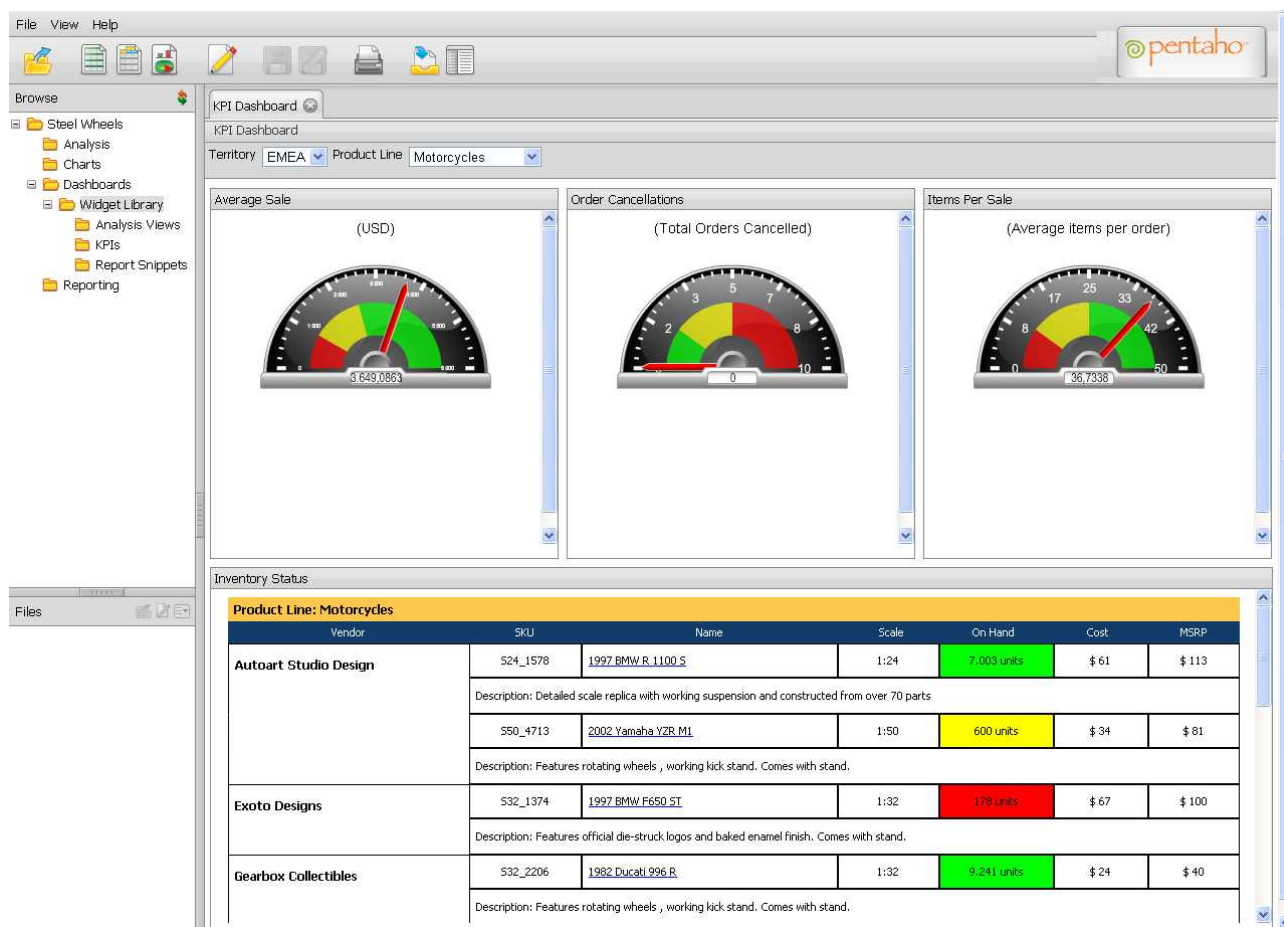
Territory	Line	2003			2004		
		Sales	Quantity	Unit Sales	Sales	Quantity	Unit Sales
APAC	Classic Cars	\$115,011	1,052	\$109	\$199,372	1,785	\$112
	Vintage Cars	\$111,639	1,243	\$90	\$147,212	1,587	\$93
	Motorcycles	\$60,789	654	\$93	\$63,159	540	\$117
	Trucks and Buses	\$11,298	91	\$124	\$80,634	801	\$101
	Planes	\$42,663	456	\$94	\$67,681	723	\$94
APAC Total		\$341,400	3,496	\$98	\$558,057	5,436	\$103
EMEA	Classic Cars	\$691,273	5,853	\$118	\$1,015,790	8,976	\$113
	Vintage Cars	\$263,695	3,094	\$85	\$504,062	5,472	\$92
	Motorcycles	\$141,836	1,428	\$99	\$204,042	2,177	\$94
	Trucks and Buses	\$228,699	2,261	\$101	\$185,421	1,558	\$119
	Planes	\$154,519	1,723	\$90	\$209,128	2,326	\$90
EMEA Total		\$1,480,021	14,359	\$103	\$2,118,443	20,509	\$103
Japan	Classic Cars	\$120,696	898	\$134	\$42,071	307	\$137
	Planes	\$60,556	677	\$89	\$49,177	547	\$90
	Trucks and Buses	\$44,498	415	\$107	\$13,349	102	\$131
	Motorcycles	\$16,485	205	\$80	\$31,959	380	\$84
	Vintage Cars	\$22,888	308	\$74	\$21,471	229	\$94
Japan Total		\$265,123	2,503	\$106	\$158,026	1,565	\$101
NA	Classic Cars	\$587,428	4,959	\$118	\$581,043	5,017	\$116
	Vintage Cars	\$281,727	3,268	\$86	\$324,815	3,576	\$91
	Motorcycles	\$178,109	1,744	\$102	\$291,421	2,809	\$104
	Trucks and Buses	\$135,936	1,289	\$105	\$252,572	2,563	\$99
	Planes	\$90,016	977	\$92	\$202,942	2,224	\$91
NA Total		\$1,273,216	12,237	\$104	\$1,652,792	16,189	\$102
Grand Total		\$3,359,761	32,595	\$103	\$4,487,319	43,699	\$103



CRIAÇÃO DE DASHBOARDS

Na criação de dashboards, a área de desenho é dividida em áreas funcionais, definidas por templates. Um utilizador escolhe os templates de quantas áreas é que deseja dividir o écran, assim como as suas posições e depois pode editar cada uma destas áreas individualmente.

Em cada área é possível adicionar ficheiros do servidor *Pentaho*, tabelas de dados ou gráficos. Sempre que adicionamos um novo elemento, somos levados para uma janela de query onde controlamos os campos que desejamos colocar no nosso elemento, assim como opções de filtragem, ordenação de dados, etc.



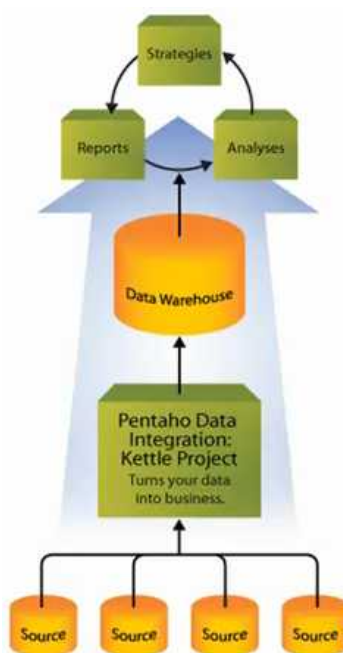


4.9 - PENTAHO DATA INTEGRATION (KETTLER)

Este software, também conhecido como KETTLE (Kettle Extraction, Transport, Transformation and Loading Environment), é uma ferramenta gratuita open source de ETL, presente na suite comercial de BI da Pentaho.

A sua característica principal é ser baseado em modelos (guardados sobre a forma de meta-dados), possuindo ferramentas de modelação alto-nível para representar transformações de dados e tarefas que queremos executar, assim como motores interpretadores desses mesmos modelos, abrangendo assim todos os processos de ETL.

Este software serve então como intermediário entre as variadas fontes de informação de uma organização para permitir a construção de um Data Warehouse centralizado, a partir do qual se podem executar variados processos de BI como mostra o esquema seguinte:

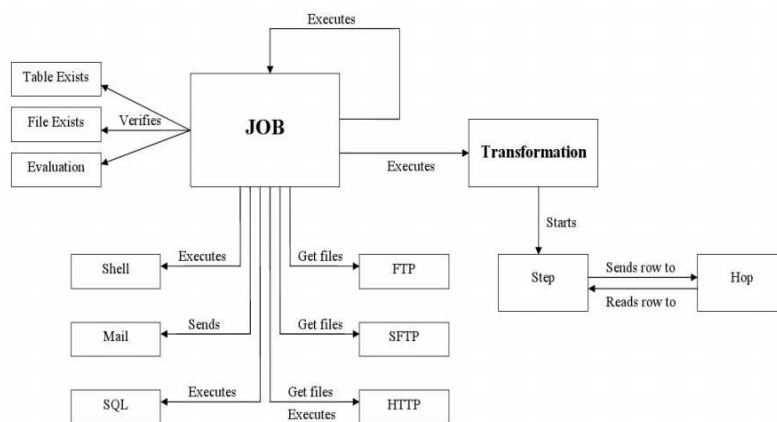


Entre as suas variadas características destaca-se o facto de ser altamente escalável, suportando a criação de clusters e de partições, paralelismo e multi-threading, assim como também tem capacidades de correr múltiplas cópias de um step numa tarefa ou transformação (um step corresponde à unidade mais básica de uma transformação ou tarefa, que explicamos mais à frente no relatório).

Possui também uma arquitectura que suporta plug-ins, podendo assim ser expandido conforme as necessidades do cliente. Todas as tarefas e transformações modeladas são dinâmicas, uma vez que usam variáveis para definir inputs e outputs, aumentando a capacidade de reutilização dos modelos criados.



Heis o modelo conceptual que representa a relação entre transformações e tarefas:



Como podemos observar, as transformações são os modelos mais baixo-nível desta ferramenta. São estes modelos que irão representar as transformações ETL efectuadas sobre as nossas fontes de informação, construindo-as através de steps. Cada step representa uma operação de transformação básica sobre os nossos dados e são interligados entre si através de fluxos de dados, transformando passo a passo a nossa informação.

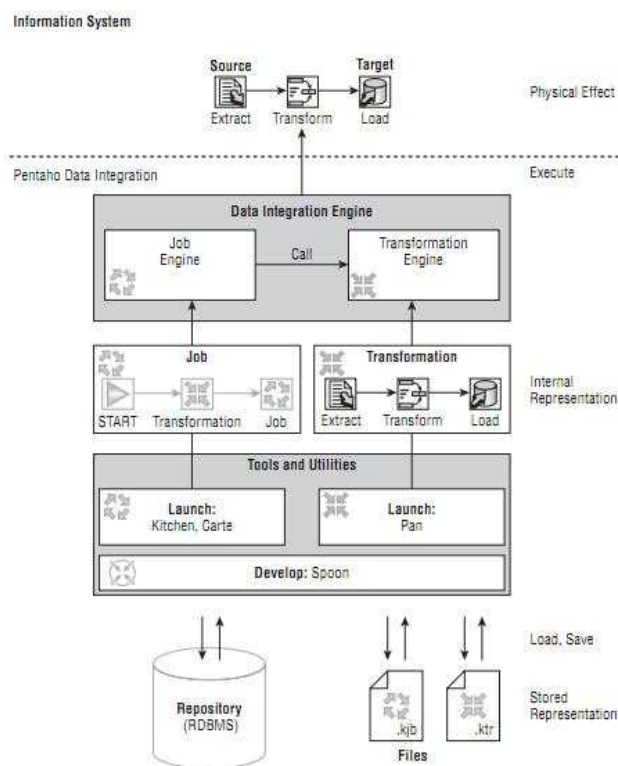
Uma vez guardadas as nossas transformações, são integradas em tarefas, cuja única função é interligar diferentes transformações entre si numa sequência lógica. Estas tarefas representam modelos mais alto-nível sobre o processo ETL e efectuam validações básicas sobre bases de dados, ficheiros, etc, assim como outras operações úteis dum ponto de vista administrativo de data warehousing, tais como envio de ficheiros por diversos protocolos, execução em shell, agendamento de transformações e outras tarefas, etc.

O *Pentaho Data Integration* é dividido em quatro ferramentas principais:

- *Spoon* – uma ferramenta de interface gráfica usada para criar transformações e tarefas, possibilitando actividades de debug, teste e previsão sobre as suas criações.
- *Carte* – um servidor ETL que permite a execução remota de transformações e tarefas.
- *Pan* – uma ferramenta de linha de comandos que executa transformações modeladas no Spoon.
- *Kitchen* – uma ferramenta de linha de comandos que executa tarefas modeladas no Spoon.
- *Pentaho Data Integration Console* – apenas presente nas versões comerciais do Pentaho, é um componente que permite a definição de alertas, armazenar logs, controlar a execução remota de tarefas e transformações, assim como guarda um conjunto de métricas relacionadas com performance.



Neste esquema podemos ver em mais detalhe os diferentes componentes do *Pentaho Data Integration*:



O *Spoon* é usado para modelar graficamente tarefas e transformações, podendo-as armazenar num repositório especializado para este tipo de modelos.

Após a modelação de transformações e de integrá-las em tarefas, estas podem ser executadas localmente através do ambiente gráfico do *Spoon*, através da aplicação de linha de comandos *Kitchen* ou então remotamente através do servidor *Carte*.

Os modelos XML das tarefas (guardados com a extensão .kjb para tarefas e .ktr para transformações) ao serem executado são interpretados por um motor de tarefas, o qual vai chamando arbitrariamente o motor de transformações à medida que estas aparecem nos seus steps. As transformações também podem ser executadas localmente através da aplicação de linha de comandos *Pan* de forma isolada.



4.10 - PRODUTOS OPEN SOURCE DA PENTAHO

Para além dos componentes open source listados na arquitectura do servidor Pentaho, a própria empresa disponibiliza os seguintes produtos open source:

- *Pentaho Data Integration (Kettle).*
- *Pentaho Reporting*, juntamente com o seu motor embebido de criação de relatórios.
- *Pentaho Server* e o *Pentaho Analyser* (Consola de utilizador).
- *Mondrian OLAP Server.*
- *Weka Data Mining.*
- *Community Dashboard Framework* – Uma framework para a criação de dashboards alternativa ao JPivot e implementada numa arquitectura de repositórios semelhante à plataforma *Pentaho*.
- *Community Data Access* – Um plug-in desenhado para a suite da *Pentaho* para estender a flexibilidade de acesso a várias fontes de dados diferentes, manobrando a necessidade de execução processos ETL prévios para conseguir angariar dados de diferentes bases de dados nas ferramentas da suite.



5 - MANUAL DE UTILIZAÇÃO DO PENTAHO DATA INTEGRATION

5.1 - ECLIPSE SETUP

No IDE do eclipse, baixamos a última versão estável do *Pentaho Data Integration*, a versão 4.0.1 (<http://sourceforge.net/projects/pentaho/files/Data%20Integration/4.0.1-stable/>) para testar este componente.

Uma vez descarregado, basta importar todos os ficheiros .jar da pasta {pdi_home}\lib e da pasta {pdi_home}\libext para começarmos a utilizar o API java do *Pentaho Data Integration*.

5.2 - CASOS DE USO – CRIAR UMA TRANSFORMAÇÃO E EXECUTÁ-LA

Imaginando que queremos extrair informação da seguinte tabela da base de dados, para a criação de um data warehouse:

id	loja	vendas	item	ano
1	Loja A	345	Portátil A	2009
2	Loja B	364	Portátil B	2010
3	Loja C	226	Portátil C	2008
4	Loja D	463	Desktop A	2009
5	Loja B	674	Desktop B	2008
6	Loja D	354	Desktop C	2010
7	Loja C	324	Portátil A	2009
8	Loja A	123	Portátil A	2010
9	Loja B	44	Portátil B	2009
10	Loja D	234	Portátil C	2010
11	Loja A	33	Desktop B	2008
12	Loja C	66	Portátil A	2008
13	Loja D	324	Desktop B	2009
14	Loja C	765	Portátil A	2008
15	Loja A	123	Portátil B	2010
16	Loja C	255	Desktop B	2009
17	Loja A	466	Portátil A	2008
18	Loja C	234	Desktop C	2010
19	Loja D	645	Portátil B	2009
20	Loja A	234	Desktop C	2009
21	Loja B	754	Portátil A	2010
22	Loja A	356	Desktop C	2009
23	Loja D	246	Portátil C	2008
24	Loja C	255	Portátil B	2010
25	Loja A	453	Portátil A	2009
26	Loja B	123	Portátil C	2008
27	Loja B	534	Portátil C	2010
28	Loja D	357	Portátil B	2009
29	Loja A	646	Portátil A	2008

Começamos por criar uma transformação simples na aplicação Spoon. Após executarmos o interface a partir de {pdi_home}\Spoon.bat irá aparecer-nos uma janela inicial, onde nos perguntam se queremos adicionar um repositório.

Este repositório serve para guardar transformações e tarefas para que possam mais tarde ser carregadas directamente da base de dados através do java, podendo facilitar futuramente a gestão de um número elevado destes modelos. No entanto, neste caso específico não o iremos usar.

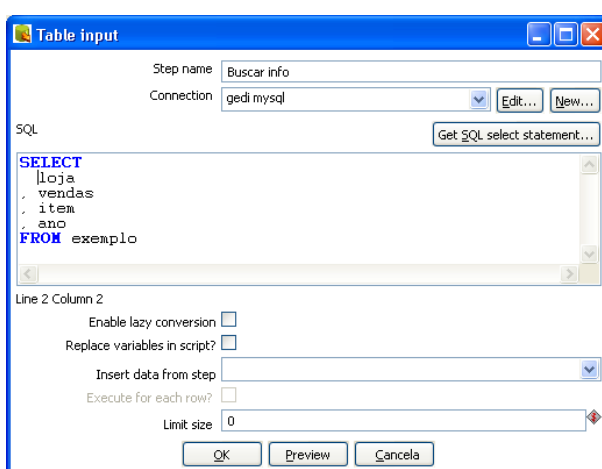
Seleccionamos **File > New > Transformation** e uma perspectiva especial de desenho de transformações irá aparecer no écran. O interface foi construído para ser drag-and-drop, contendo todos os steps possíveis de adicionar a uma transformação do lado direito do écran e uma área de desenho central. No nosso exemplo, iremos expandir a opção **Input** onde estão localizadas todas as opções disponibilizadas de leitura de fontes externas de informação. Arrastamos um step de **Table input** para acedermos à tabela na base de dados, a partir da qual iremos criar um data warehouse com várias dimensões.



Ao seleccionarmos este novo componente com o botão direito do rato e escolhermos **Edit** (ou carregarmos duas vezes com o rato sobre o step) deparamo-nos com uma nova janela onde iremos escolher a query que queremos fazer sobre a base de dados. No entanto, precisamos de primeiro adicioná-la. Para o fazer, perto da caixa de texto **Connection** seleccionamos **New** e aparecerá uma outra janela onde podemos colocar não só as credenciais de acesso e tipo de base de dados que vamos usar, como também opções específicas à base de dados escolhida.

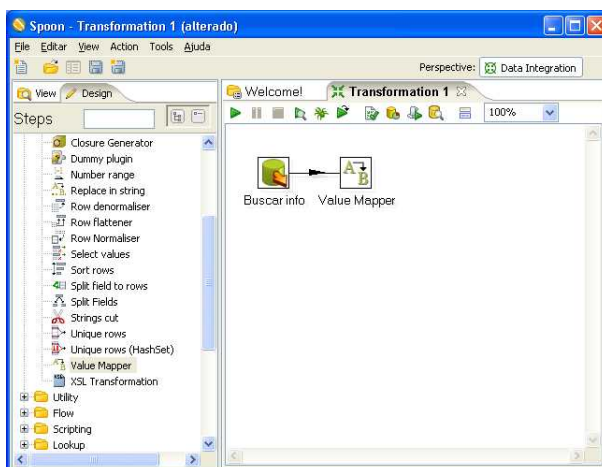
Uma vez adicionada a base de dados, ao seleccionarmos **Get SQL select statement** conseguimos escolher de quais as tabelas é que queremos extrair dados, embora a query possa ser escrita por extenso na janela de edição do step **Table input**. Também podemos limitar o número de resultados que queremos no fundo da janela, sendo útil para pequenos testes antes de utilizar um exemplo realista numa transformação com dezenas de milhares de entradas nas suas tabelas.

Também podemos seleccionar preview para comprovar que os dados estão a ser extraídos correctamente.



Seguidamente criamos um step **Value Mapper** (presente no menu drop-down **Transform**) na área de desenho para criar duas novas categorias onde iremos agregar produtos: Desktops e Portáteis. Estas categorias servirão mais tarde para criar dimensões.

Para podermos passar os dados extraídos do primeiro step de leitura da base de dados para o novo step precisamos de clicar sobre o primeiro step com o botão esquerdo do rato, mantendo a tecla shift pressionada, terminando ao clicar sobre o novo step. Isto irá criar uma seta de transição de dados entre os dois elementos, representando um fluxo de dados:





No novo step, ao editá-lo, dizemos qual o campo a partir do qual queremos mapear valores em Fieldname to use. Colocamos de seguida o nome da nova coluna onde os novos valores mapeados irão ser armazenados (nesta caso designa-mo-la por “categoria”).

Finalmente mapeamos cada um dos nomes dos produtos (que escrevemos em Source value) para a sua nova categoria (Target value). Podemos também fazer preview no step seleccionado carregando no ícone de preview presente acima da área de desenho de transformações, retornando uma grelha com os valores extraídos, seguidos da nova coluna “categoria”.

Value Mapper

Step name : Criar categorias

Fieldname to use : Item

Target field name (empty=overwrite) : categoria

Default upon non-matching :

Field values:

	Source value	Target value
1	Portátil A	Portátil
2	Portátil B	Portátil
3	Portátil C	Portátil
4	Desktop A	Desktop
5	Desktop B	Desktop
6	Desktop C	Desktop
7		

OK Cancela

Podemos agora criar uma nova transformação de dados de maneira a ajustar o nome das colunas, assim como outras operações relacionadas com manipulação de dados. Para este processo criamos um step **Select values** (presente no menu drop-down **Transform**) e ligamo-lo ao step anterior como fonte de dados, conforme exemplificado anteriormente. Ao editá-lo, procedemos ao ir para o separador **Meta-data** e seleccionamos a opção **Get fields to change** para carregar as colunas na grelha da janela de edição. Aqui é que seleccionamos os novos nomes dos campos, assim como podemos especificar tipos de dados, tamanho, formato, etc.

Select / Rename values

Step name : Transformar dados

Select & Alter / Remove / Meta-data

Fields to alter the meta-data for :

	Fieldname	Rename to	Type	Length	Precision	Binary to Normal?	Format	Encoding	Decimal	Grouping	Currency
1	loja	Lojas	String	6		N					
2	vendas	Vendas	Integer	10	0	N					
3	item	Produto	String	20		N					
4	ano	Ano	Integer	4	0	N					
5	categoria	Categoria	String	10		N					

Get fields to change

OK Cancela

Uma vez terminada a nossa manipulação de dados, estamos prontos para começar a criar o nosso data warehouse. Podemos fazer este passo começando por criar dimensões, finalizando pela criação da tabela de factos. Para criar a nossa primeira dimensão, criamos um step **Combination lookup/update** (presente no menu drop-down **Data warehouse**) e interligamos ao step anterior.



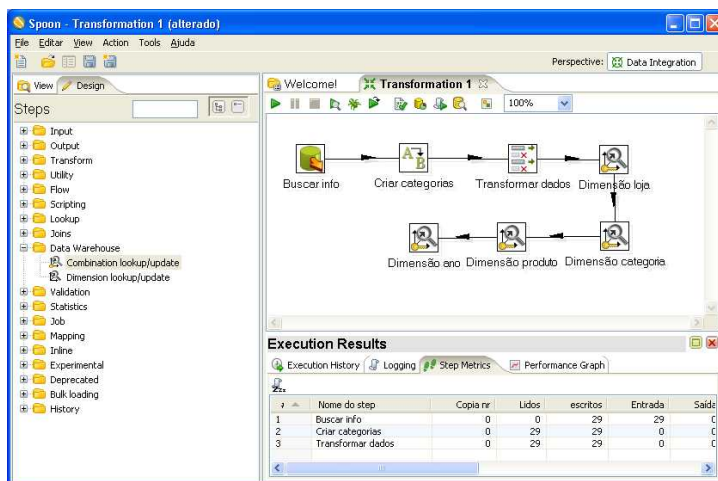
Na sua edição, colocamos como tabela de destino “dimensao_computadores_loja” e no campo de chave técnica colocamos “id_loja”. Selecionamos a opção de **Usa o campo de auto incremento** para incrementar índices da chave primária automaticamente, acabando por seleccionar também **Remove campos lookup** para irmos filtrando os dados, até só sobrarem chaves para usar na tabela de factos. Por fim carregamos no botão **Obtém campos** para carregar todos os campos disponíveis dos dados extraídos na grelha e apagamos todos excepto o campo “Lojas” .

No final desta operação obtemos uma janela igual a esta:

Para criar a tabela na base de dados a partir do Spoon, necessitamos de carregar no botão **SQL**. A partir desta nova janela podemos ver a query SQL gerada automaticamente pelo Spoon que permitirá criar a nossa primeira tabela de dimensão. Para executar a query SQL gerada basta seleccionar **Execute** e o Spoon retorna o resultado de execução da query.



Repetimos este passo para criar a dimensão categoria, dimensão produto, dimensão ano, interligando sempre o step anterior, até obtermos um diagrama igual ao seguinte:



Para finalizar a transformação, necessitamos sempre de definir um output. Neste caso iremos terminar com a criação da tabela de factos do nosso data warehouse através do componente **Table output** (presente no menu drop-down **Output**). Na edição do componente, basta colocar o nome da tabela, uma vez que já filtrámos toda a informação relevante para a tabela de factos nos steps anteriores.

Nome do Step: Criar tabela factos

Connection: gedi mysql

Target schema:

Target table: tabela_factos_computadores

Commit size: 1000

Truncate table: ☐

Ignore insert errors: ☐

Specify database fields: ☐

Main options Database fields

Partition data over tables: ☐

Partitioning field:

Partition data per month: ☒

Partition data per day: ☐

Use batch update for inserts: ☒

Is the name of the table defined in a field?: ☐

Field that contains name of table:

Store the tablename field: ☒

Return auto-generated key: ☐


Name of auto-generated key field:

OK Cancela SQL

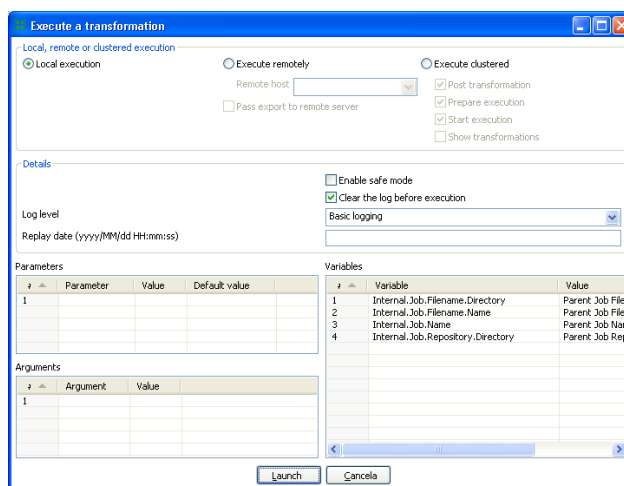
Neste momento podemos guardar esta transformação em dois formatos possíveis, com a extensão **ktr** ou em **xml**, conforme os nossos requisitos. Ambos os formatos podem ser executados a partir do API java sem qualquer restrição, assim como a partir da interface gráfica do Spoon ou remotamente através do servidor Carte.



Após termos guardado a nossa transformação, torna-se simples executá-la directamente através do Spoon.

Podemos fazer isto carregando no ícone  localizado na barra directamente acima da área de desenho das transformações. Depois de pressionado, aparece-nos uma nova janela com várias opções de execução, entre elas a execução remota por um servidor Carte. No entanto, tendo em conta que a opção de execução directa no Spoon está seleccionada por omissão, basta pressionar o botão **Launch** e a transformação é lançada.

Durante a execução também é útil o separador **Step metrics** localizado na parte inferior da área de desenho que nos permite verificar determinadas estatísticas sobre a execução, incluindo erros que tenham ocorrido, quantidade de linhas percorridos em cada step, etc.



Outra opção seria recorrermos ao *Pan*, a alternativa de linha de comandos. Uma vez dentro da directoria do pacote do *Pentaho Data Integration*, podemos executar a nossa transformação da maneira seguinte:

```
pan.bat /file:"C:\Transformations\criar_dw.ktr" /level:Basic
```

Ou então, em sistema unix:

```
pan.sh -file:"/Transformations/criar_dw.ktr" -level=Minimal
```

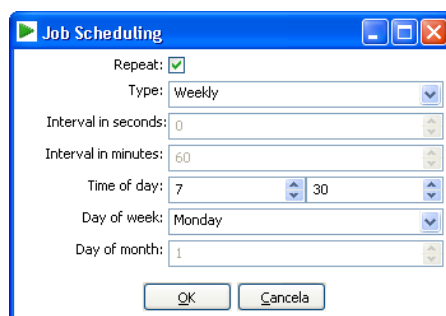


5.3 - CASOS DE USO – CRIAR UMA TAREFA E EXECUTÁ-LA

A criação de uma tarefa é semelhante à criação de transformações, tendo em conta que é feito na mesma interface drag-and-drop que usámos anteriormente. Com a aplicação Spoon iniciada, seleccionamos **File > New > Job** e uma perspectiva especial de desenho de tarefas irá aparecer no écran.

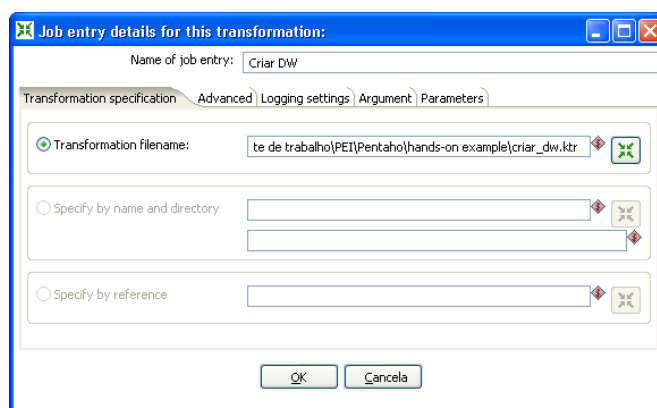
Iniciamos sempre qualquer tarefa com o componente **START** (presente no menu drop-down **General**). Ao abrirmos o seu respectivo painel de edição, podemos controlar parâmetros de agendamento de execução, regularmente presentes no back-end por detrás de um data warehouse.

A figura seguinte mostra uma configuração exemplo de execução regular semanal, às 7:30 de todas as segundas-feiras:



Depois de termos definido o agendamento da nossa tarefa, podemos interligar este step a uma transformação ou até mesmo a uma outra tarefa. No nosso caso iremos interligar este step (com o método descrito previamente na criação de transformações) a um step **Transformation** presente no mesmo menu do **START**.

No seu modo de edição, escolhemos qual o ficheiro que contém a transformação que queremos executar no campo **Transformation filename**. Podemos também através dos outros separadores controlar mais parâmetros e opções que podem ser úteis, tais como capacidade de logging da execução da transformação para um ficheiro de texto à escolha. Neste exemplo colocámos o ficheiro criado na transformação exemplo criada previamente:

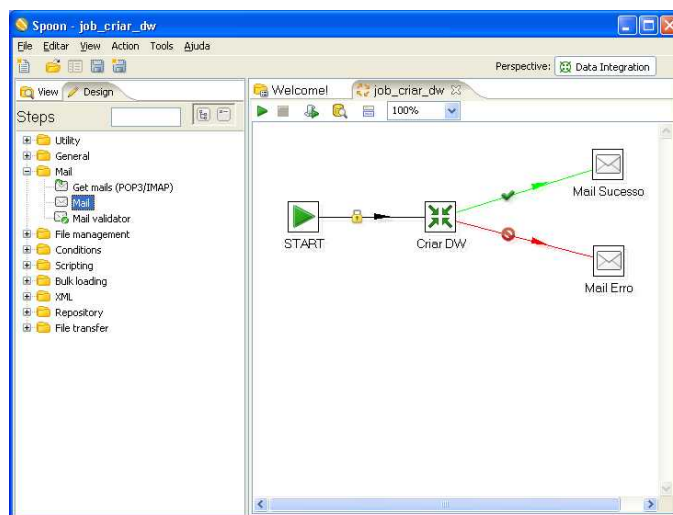


Agora vamos mostrar outra funcionalidade interessante das tarefas do Kettle em termos de gestão.

Para isso iremos arrastar dois componentes **Mail** para a área de desenho. Não iremos explicar os detalhes da edição deste componente visto que é bastante directa.



Ao interligarmos o step da transformação com um dos componentes **Mail** obtemos um arco de avaliação. Neste tipo de arco/fluxo de dados ocorre uma avaliação sobre a execução do step anterior, podendo representar um caso de sucesso (por omissão é o primeiro arco que aparece quando se faz a primeira interligação a partir duma transformação ou tarefa) ou um caso de erro. Isto permite-nos criar uma gestão de alto nível de qualquer transformação ou tarefa criada no Kettle, interessante em termos administrativos.



Após executarmos esta tarefa, ela irá ficar permanentemente em espera dentro do Spoon, executando no intervalo anteriormente especificado. Outra alternativa para executar tarefas sem o Spoon aberto é através do Kitchen, a alternativa de linha de comandos disponibilizada no *Pentaho Data Integration*.

Para tal, basta executar o Kitchen com os seguintes parâmetros:

```
kitchen.bat /file:"C:\Jobs\job_criar_dw.kjb" /level:Basic
```

Ou então em ambiente unix:

```
kitchen.sh -file="/Jobs/job_criar_dw.kjb" --level=Minimal
```

Outra possibilidade seria executar a partir de um repositório de transformações e tarefas:

```
kitchen.bat /rep:"repositorio" /job:"a minha tarefa" /dir:/Dimensoes  
/user:meu_username /pass:minha_password /level:Basic
```




5.4 - CASOS DE USO – EXECUTAR UMA TRANSFORMAÇÃO NO JAVA API

Após guardarmos a transformação anterior, podemos executá-la através do seguinte trecho de código java:

```
public static void runTransformation(String filename) {
    try {
        KettleEnvironment.init();
        EnvUtil.environmentInit();
        TransMeta transMeta = new TransMeta(filename);
        Trans trans = new Trans(transMeta);
        trans.execute(null); // Consegue-se passar argumentos aqui
        trans.waitUntilFinished();
        if ( trans.getErrors() > 0 ) {
            throw new RuntimeException( "Erro na execução" );
        }
    } catch ( KettleException e ) {
        System.out.println(e);
    }
}
```

Outra opção que existe e que pode ser útil em termos de reutilização das transformações passa por definirmos variáveis quando criamos uma transformação. Campos que possam ser parametrizados com variáveis estão marcados no ambiente gráfico do Spoon com o ícone

Também podemos usar esta funcionalidade, por exemplo, ao colocarmos a seguinte query no editor SQL do Spoon:

```
SELECT item FROM exemplo LIMIT 0, ${QUERY_LIMIT}
```

Depois da transformação ter sido guardada e carregada no java com o método descrito acima, é possível alterar a variável QUERY_LIMIT dinamicamente antes de executarmos a transformação, recorrendo ao código auxiliar seguinte:

```
trans.setParameterValue("QUERY_LIMIT", "15");
```



5.5 - CASOS DE USO – EXECUTAR UMA TAREFA NO JAVA API

Semelhante à execução de transformações, a execução de tarefas pode ser conseguida através do seguinte trecho de código:

```
public static void runJob(String filename) throws KettleException {
    KettleEnvironment.init();
    EnvUtil.environmentInit();
    // Segundo argumento pode ser um repositório
    JobMeta jobMeta = new JobMeta(filename, null);
    // Primeiro argumento pode ser um repositório
    Job job = new Job(null, jobMeta);
    // Inicia argumentos e variáveis internas
    job.getJobMeta().setArguments(null);
    job.initializeVariablesFrom(null);
    job.getJobMeta().setInternalKettleVariables(job);
    // Percorremos todos os parâmetros necessários e passamo-los para
    // JobMeta dentro do objecto Job
    String[] jobParams = jobMeta.listParameters();
    for (String param : jobParams) {
        job.getJobMeta().setParameterValue(param, "exemplo_parametro");
    }
    job.copyParametersFrom(job.getJobMeta());
    // Activamos os novos parâmetros
    job.activateParameters();
    // Corremos a tarefa
    job.start();
    // Manutenção até a tarefa terminar
    job.waitUntilFinished();
    // Se necessário, obtemos o resultado da execução da nossa tarefa
    Result result = job.getResult();
    if (job.getErrors() > 0) {
        throw new RuntimeException("Erro ao executar tarefa");
    }
}
```



5.6 - CASOS DE USO – CONFIGURAR E LANÇAR O SERVIDOR CARTE MANUALMENTE

Para configurar este servidor basta definirmos um ou mais utilizadores autorizados a acedê-lo. Configuramos os utilizadores editando o ficheiro `{pdi_home}\pwd\kettle.pwd`. O username e password por omissão é *cluster*, embora devamos apagar este utilizador do ficheiro de configuração por razões de segurança.

Podemos também optar por ofuscar a password no ficheiro de configuração recorrendo a um script disponibilizado em conjunto com o pacote do *Pentaho Data Integration*, o *Encr.bat*. Depois é só copiar a string gerada para o ficheiro de configuração, em vez da password em texto aberto.

Uma chamada típica deste script seria:

```
Encr.bat -carte a_minha_password
```

Para executar o Carte manualmente:

```
Carte.bat ip_servidor porto
```



5.7 - CASOS DE USO – EXECUTAR UMA TRANSFORMAÇÃO REMOTAMENTE NO CARTE

Começamos inicialmente por lançar o servidor Carte a partir do java para posteriormente executar a nossa tarefa ou transformação.

Para tal, temos de definir a nossa configuração de slave servers (visto que podemos utilizar também configurações com clusters de slave servers, embora não iremos cobrir essa funcionalidade neste relatório) e depois passá-la como argumento para a classe Carte.

Heis um exemplo de como lançar o servidor Carte com apenas um slave server para executar pedidos:

```
public static void runCarte() {
    // Cria o servidor Carte a.k.a. Slave Server
    SlaveServer ss = new SlaveServer("nome_servidor", "ip_servidor",
        "porto", "username", "password");
    // Criamos uma config simples de apenas um slave server
    SlaveServerConfig ssc = new SlaveServerConfig(ss);
    try {
        Carte.runCarte(ssc);
    } catch (Exception e) {
        // Erro ao lancar o servidor
        e.printStackTrace();
    }
}
```

Lançado o servidor, temos de criar uma configuração de transformação, através da qual definimos o servidor (ou servidores, dependendo da arquitectura de slave servers que escolhemos) que irá executar a transformação. Depois de definida a configuração, enviamos para o API do Kettle a transformação desejada, juntamente com a configuração para saber para qual servidor é que a deve enviar para ser executada:

```
public static void runTransformationRemotely(String filename) {
    try {
        KettleEnvironment.init();
        EnvUtil.environmentInit();
        TransMeta transMeta = new TransMeta(filename);
        Trans trans = new Trans(transMeta);
        // Criamos uma config de execucao de transformacoes
        TransExecutionConfiguration tc = new TransExecutionConfiguration();
        // Cria o servidor Carte a.k.a. Slave Server
        SlaveServer ss = new SlaveServer("nome_servidor", "ip_servidor",
            "porto", "username", "password");
        // Atribuimos o slave server ah configuracao
        tc.setRemoteServer(ss);
        // Enviamos para o servidor a transformacao
        Trans.sendToSlaveServer(transMeta, tc, null);
        trans.execute(null);
        trans.waitUntilFinished();
        if ( trans.getErrors() > 0 ) {
            throw new RuntimeException( "Erro na execução" );
        }
    } catch ( KettleException e ) { System.out.println(e); } }
```



Uma questão importante, no que toca a execuções remotas de tarefas, é o uso de caminhos para ficheiros locais (sub-transformações ou até mesmo outras tarefas). Muitas vezes existem conflitos ao enviar uma tarefa com um caminho local para um servidor remoto, perdendo a referência e impedindo a execução correcta da mesma. Existem algumas maneiras de contornar este problema:

- Usar URLs nos campos de caminhos para ficheiros.
- Usar um caminho absoluto nas transformações do lado do servidor (copiando os ficheiros respectivos previamente).
- Usar o método `setPassingExport(true)` dentro do `JobExecutionConfiguration` antes de enviar a tarefa para o servidor, forçando-o a exportar a tarefa e todas as suas sub-tarefas e sub-transformações dentro do ficheiro .zip, executando-o dentro do mesmo.

Um dos autores do PDI chega mesmo a sugerir o uso de um sistema de controlo de versões como o *Subversion* para gerir as transformações e tarefas no servidor, agendando actualizações automáticas e gerindo as alterações feitas ao longo do tempo sobre elas.



6 - OUTRAS FRAMEWORKS ETL OPEN SOURCE

Após termos analisado o *Pentaho Data Integration*, podemos proceder à comparação de outros produtos open source de ETL. Para esta análise, escolhemos apenas dar oportunidade aos projectos mais influentes e usados na actualidade, tendo também em conta a sua maturidade.

Considerámos como possíveis componentes alternativos as frameworks seguintes (por ordem de popularidade):

- Talend Open Studio
- CloverETL

6.1 - TALEND OPEN STUDIO

O *Talend Open Studio* é considerado o maior competidor directo do *Pentaho*, sendo o segundo maior produto open source de ETL presente actualmente no mercado.

Este software apresenta uma estrutura bastante diferente do *Pentaho Data Integration*, tendo em conta que não se baseia em motores interpretadores de modelos. Este software opera internamente como um gerador de código, podendo gerar código java e perl para executar os seus sub-processos de ETL.

A sua interface gráfica também é diferente, assim como o seu modelo geral de funcionamento, considerando que este estúdio organiza todo o seu arsenal num workspace colaborativo, baseando-se na interface do eclipse.

Dentro desta interface existe um repositório de meta-dados, usado para armazenar definições e configurações para cada tarefa, e uma área de desenho gráfico de tarefas (*jobs*) orientada a componentes semelhante à alternativa da Pentaho. As tarefas depois de criadas são exportadas para um ficheiro .zip que contém um executável .bat que invoca todo o código gerado pelo nosso modelo.

Também apresenta a possibilidade de criar modelos alto-nível de negócio, sendo útil para manter todos os intervenientes no processo de data warehousing a par do que é necessário fazer (embora não haja qualquer interligação com as tarefas desenhadas, mantendo um carácter meramente informativo).

Para analisar esta ferramenta devidamente recolhemos algumas opiniões e factos relevantes de comparações disponíveis na internet, assim como decidimos replicar o mesmo exemplo usado no *Pentaho Data Integration* para testar o seu modo de funcionamento mais a nível de usabilidade, disponibilidade de documentação e curva de aprendizagem necessária.

REPLICANDO O EXEMPLO

Para replicarmos o exemplo anterior foi necessário bastante esforço da nossa parte para tentar efectuar as mesmas operações essenciais de data warehousing.

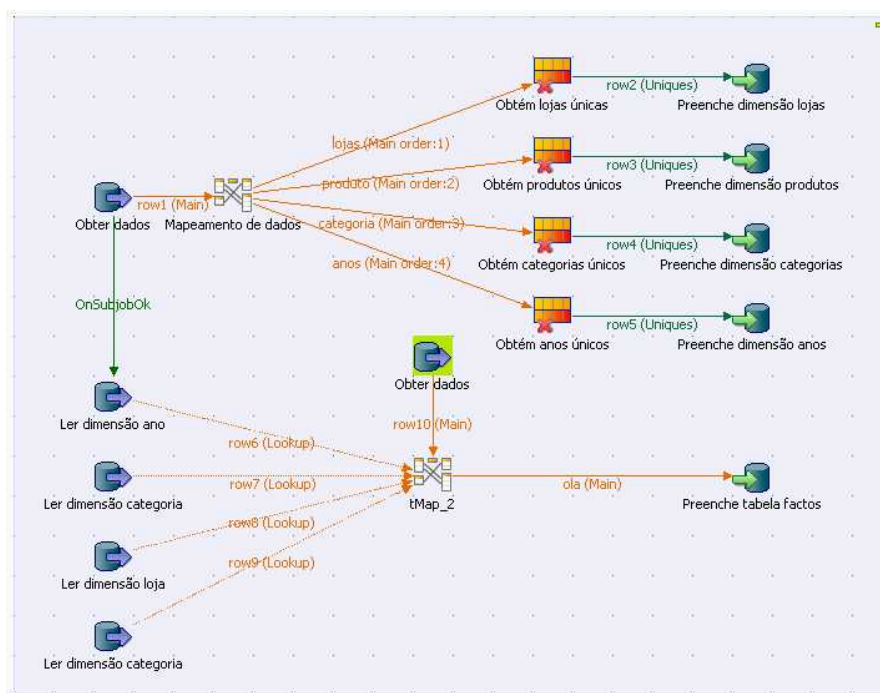
Os developers do Talend disponibilizam uma colecção bastante extensa de documentação, entre eles um manual de uso do *Open Studio*, um manual com documentação referente a todos os componentes gráficos disponíveis para modelar as nossas tarefas, um wiki que serve como knowledge-base, um fórum para a comunidade e ainda tutoriais básicos de vídeo

No entanto, sentimos que a curva de aprendizagem da ferramenta é demasiado acentuada para que chegue a compensar o tempo despendido a criar um modelo de tarefa, mesmo com a suposta grande quantidade de documentação.



Os tutoriais são demasiado básicos considerando a abordagem de baixo-nível de cada componente, a documentação dos componentes chega mesmo, em parte, inútil devido à sua complexidade e falta de exemplos na internet do seu uso. Mesmo algo tão simples como a criação de uma tabela numa base de dados MySQL com uma chave primária automaticamente -incrementada torna-se um problema complexo de se resolver dentro desta ferramenta.

É verdade que esta ferramenta ganha muitos pontos a favor em termos de granularidade sobre o que queremos fazer, podendo apresentar operações complexas de modelação de dados e interacção com a plataforma de business intelligence. De qualquer maneira, para fazermos uma comparação com o *Pentaho*, em termos de modelação foram necessários 17 componentes gráficos para apenas representar o preenchimento de dados do data warehouse no nosso exemplo. E isto excluindo a criação das tabelas em si.



Outra nota importante sobre o uso deste software foram alguns bugs encontrados durante o seu uso. Notámos a permanência de componentes gráficos na área de desenho impossíveis de apagar (e não listados nas outras perspectivas do workspace) e que por vezes desapareciam.

Outro bug fatal encontrado foi durante o encerramento da nossa máquina de testes sem fechar previamente a aplicação. O *Open Studio* não guardou devidamente as definições do workspace que estávamos a usar, corrompendo o modelo que estávamos a desenvolver até então, assim como todo o workspace e interface gráfica.

Apesar da sua dificuldade de utilização, da presença de alguns bugs pontuais, e de ser ligeiramente menos poderosa em termos de performance em comparação com o *Pentaho Data Integration*, acreditamos que possa ser usado em transformações de dados mais granulares e complexas.



PRINCIPAIS DESTAQUES

Iremos listar então alguns destaques sobre esta ferramenta, positivos e negativos, que iremos considerar depois para a comparação:

- Geração de código no background feita através do Perl.
- Bastante granular nas transformações moduladas.
- Elevado número de componentes complexos e completamente configuráveis.
- Possibilidade de usar scripts em java ou perl para complementar determinados passos na transformação de dados.
- Código gerado pela ferramenta pode ser modificado.
- Projecto relativamente imaturo (com a sua primeira milestone lançada em 2006).
- Suporte para trabalho colaborativo através de repositórios.
- Uso de meta-dados permite reutilização de componentes entre várias tarefas.
- Suporte para criação e uso de plug-ins
- Multi-plataforma, tendo em conta que é baseado em java.
- Documentação prestada por vezes inútil.
- Auxílio do code-completion típico de um ambiente gráfico eclipse.
- Boa capacidade de controlo sobre o logging de execução de tarefas.
- Paralelismo fraco nas versões gratuitas open-source.
- Boa capacidade de debug, uma vez que é baseado na perspectiva debug do eclipse.
- Geração automática de documentação em HTML para cada transformação modelada
- Ambiente de trabalho lento a iniciar.
- Evolução rápida da ferramenta em comparação à alternativa da *Pentaho*.
 - Injecção de capital de duas empresas financeiras.
 - Apenas focado na componente ETL.
 - Avançou da versão 4.0.1 para a versão 4.0.3 durante a escrita deste relatório.



6.2 - CLOVERETL

O *CloverETL* é uma ferramenta que considerámos como uma alternativa viável de transformações ETL devido ao seu rápido crescimento nos últimos anos a nível de funcionalidades implementadas e expansão da comunidade por detrás do projecto.

Apresenta algumas características em comum com o *Pentaho Data Integration*, tendo em conta que é também baseado em motores interpretadores de modelos. Estes modelos também são similarmente criados através duma interface gráfica, a qual também é baseada no eclipse.

Em termos de funcionalidades, apresenta um conceito que aparenta ser uma tentativa de mistura da facilidade de utilização do *Pentaho Data Integration* na modelação de transformações, em conjunto com a possibilidade de inserção de código customizável nos seus componentes como o *Talend Open Studio*, conseguido através duma linguagem específica do motor do *CloverETL*: o *CloverETL Transformation Language (CTL2)*.

O seu modo de funcionamento é inteiramente baseado em meta-dados, quer nos componentes em si, quer nas ligações de dados entre eles, exponenciando ao máximo o factor de reutilização das transformações criadas.

Apesar de apresentar esta combinação interessante de funcionalidades, a sua versão gratuita ficou um bocado aquém das nossas perspectivas, faltando bastantes componentes de design de transformações, assim como outras capacidades interessantes de paralelismo e execução remota presentes nas versões comerciais do produto. Os modelos criados através da versão gratuita também são limitados a 20 componentes, tornando-se inútil na perspectiva do nosso projecto.

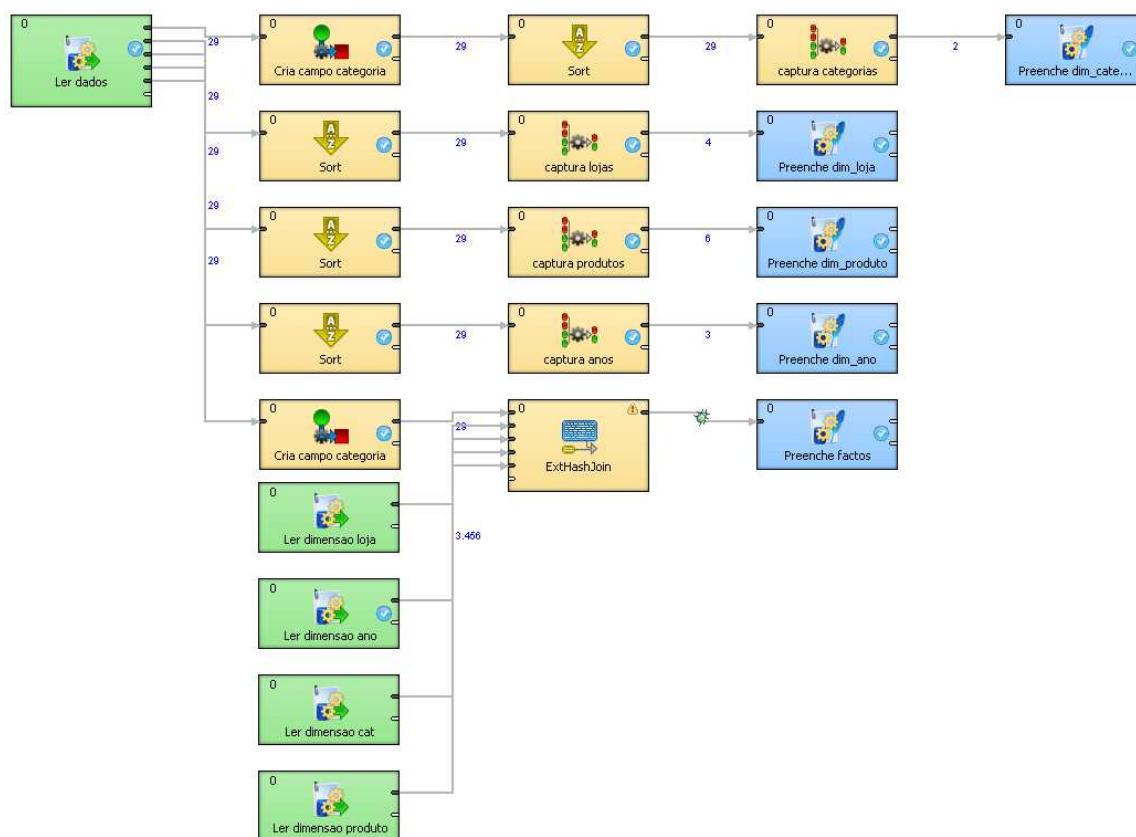
REPLICANDO O EXEMPLO

A abordagem que tivemos sobre o *CloverETL Community Edition* foi bastante satisfatória num ponto de vista geral. A curva de aprendizagem deste software assemelha-se bastante à do *Pentaho Data Integration*, considerando a simplicidade do designer gráfico, combinada com a presença abundante de documentação, exemplos e ajuda online.

Reparámos na presença de alguns bugs a nível de integridade da interface (presença de componentes e ligações que já tínhamos apagado da área de desenho), embora não sejam graves ao ponto de afectar o funcionamento do seu núcleo. Também nos deparámos com outro erro ao salvar a transformação de dados que construímos na interface do eclipse, embora a nível de funcionamento também não se tenha reflectido nada de aparente.

Este conjunto de pequenos bugs leva-nos a crer que possivelmente não haja um controlo de qualidade tão grande quanto o efectuado nas versões pagas, tendo em conta que não existem muitas queixas online sobre este produto. No fundo leva-nos a crer que esta versão funciona como um pequeno teste da simplicidade de uso, apenas para passar para as versões pagas melhoradas onde efectivamente existe um grande controle de qualidade e com suporte a funcionalidades avançadas, tais como a execução em paralelo.

Uma das limitações com que tivemos de lidar foi a ausência de componentes específicos ao processo ETL, assim como outras funcionalidades base presentes noutras suites que testámos (envio de mails para casos de sucesso e de erro, ler e escrever uma grande variedade de ficheiros, componentes de controlo de fluxo de dados, etc.).



Apesar de termos conseguido replicar aos poucos (e apenas parcialmente) o nosso exemplo, notámos que tivemos de abordar estratégias alternativas para contornar esta limitação de funcionalidades, exigindo um maior esforço de nossa parte ao modelar uma transformação que, com os componentes presentes nas versões pagas, demoraria uma fracção do tempo e dos componentes usados, forçando sobre o modelo uma complexidade desnecessária que acreditamos que em processos maiores possa afectar a performance do mesmo.



PRINCIPAIS DESTAQUES

Iremos listar então alguns destaques sobre esta ferramenta, positivos e negativos, que iremos considerar depois para a comparação:

- Projecto maduro (com a sua primeira milestone lançada em 2002).
- Intuitivo no seu uso através da interface gráfica.
- Presença abundante de documentação, exemplos e ajuda online.
- Pode ser embebido em aplicações java, podendo até criar modelos através de código.
- Componentes muito limitados na sua versão gratuita, tornando-se inútil no âmbito do nosso projecto.
- Possibilidade de recorrer ao CloverETL Transformation Language para auxiliar em transformações complexas.
- Ausência de funcionalidades como execução remota e paralelismo.
- Uso de meta-dados permite reutilização de componentes entre várias tarefas.
- Suporte para criação e uso de plug-ins
- Multi-plataforma, tendo em conta que é baseado em java.
- Suporta pouca variedade de bases de dados e ficheiros.



7 - CONCLUSÃO

Após uma análise subjectiva da *Pentaho BI Suite*, podemos dizer que esta suite está bem preparada para cumprir qualquer requisito de business intelligence, apresentando poucas falhas a nível de funcionalidades.

É completa, estável, madura a nível de projecto, focada num sector de mercado aliciante, o agile BI, e os seus developers estão atentos às novas tendências do mundo do business intelligence, assim como estão atentos às opiniões da comunidade open source como forma constante de melhoramento.

Os seus produtos apenas estão limitados a nível de rapidez de evolução, ficando em desvantagem em comparação, por exemplo, com a *Talend* que tem apresentado um rápido crescimento devido ao constante investimento por parte de terceiros, embora pertençam a sectores de mercado ligeiramente diferentes, sendo a *Talend* uma empresa com um produto apenas focado no ETL e a *Pentaho* uma empresa mais abrangente, procurando disponibilizar soluções para toda a problemática dos processos de BI empresariais.

Em relação às frameworks open source de ETL à nossa disposição, concluímos que deve ser usado o *Pentaho Data Integration* para ser integrado no SIAG-AP na nossa nova framework de business performance.

Esta escolha deve-se em grande parte à sua modelação ágil de transformações de dados, à fácil integração com aplicações java e também à sua capacidade de execução remota e em paralelo (atingindo uma maior performance para grandes volumes de dados), tornando-se adequado ao nosso projecto no sentido em que se revelou como uma das alternativas gratuitas mais completas a nível de funcionalidades.

Esta possibilidade de elevada escalabilidade através do uso de clusters de servidores para grandes transformações de dados pode vir a fazer diferença no futuro, considerando que o nosso produto está em crescimento constante e pode vir a ser uma funcionalidade decisiva para alguns dos nossos clientes.

Não podemos esquecer também de outros factores influentes como a sua versatilidade, assim como uma presença online mais robusta a nível de comunidade, disponibilizando um maior suporte no desenvolvimento da nossa solução.



8 - REFERÊNCIAS IMPORTANTES

ODS:

- http://en.wikipedia.org/wiki/Operational_data_store
- <http://searchoracle.techtarget.com/definition/operational-data-store>

Business Intelligence:

- http://en.wikipedia.org/wiki/Business_intelligence

Business Performance Management:

- http://en.wikipedia.org/wiki/Business_performance_management

Comparações de ferramentas ETL:

- <http://it.toolbox.com/blogs/infosphere/my-review-of-the-etl-vendor-comparison-report-from-wwwetltoolcom-33310>
- <http://churriwifi.wordpress.com/2010/06/01/comparing-talend-open-studio-and-pentaho-data-integration-kettle/>
- <http://it.toolbox.com/blogs/infosphere/wiki-wednesday-comparing-talend-and-pentaho-kettle-open-source-etl-tools-16294>
- <http://searchoracle.techtarget.com/answer/Tips-for-the-ETL-tool-evaluation-and-comparison-process>
- http://www.adeptia.com/products/etl_vendor_comparison.html
- <http://www.gartner.com/technology/media-products/reprints/oracle/article109/article109.html>
- <http://www.ibridge.be/?p=150>
- <http://www.ibridge.be/?p=178>
- <http://www.information-management.com/news/1044005-1.html>

Data Warehouse:

- http://en.wikipedia.org/wiki/Data_Warehouse
- http://etl-tools.info/en/bi/datawarehouse_concepts.htm
- http://imasters.uol.com.br/artigo/11178/gerencia/construcao_de_data_warehouse_dw_e_data_mart_dm/
- <http://tdwi.org/Articles/2003/05/12/Four-Ways-to-Build-a-Data-Warehouse.aspx?Page=1>
- <http://www.exforsys.com/tutorials/msas/data-warehouse-design-kimball-vs-inmon.html>
- <http://www.information-management.com/issues/20000301/1953-1.html>
- http://www.itweb.co.za/index.php?option=com_content&view=article&id=17259:hybrid-data-warehouse-architectures&catid=112:enterprise-solutions

Extract, transform and load (ETL):

- http://en.wikipedia.org/wiki/Extract,_transform,_load
- <http://searchoracle.techtarget.com/answer/Engine-based-vs-code-generating-ETL-tools>



- http://www.learn-datamodeling.com/tm_code_generator.htm
- <http://www.manageability.org/blog/stuff/open-source-etl>

Ferramentas ETL open source:

- <http://it.toolbox.com/blogs/opensource-unleashed/open-source-catalogue-etl-9191>
- [http://sourceforge.net/softwaremap/?&fq\[\]=trove:580&fq\[\]=trove:792](http://sourceforge.net/softwaremap/?&fq[]=trove:580&fq[]=trove:792)

OLAP:

- <http://data-warehouses.net/glossary/conformeddimensions.html>
- http://en.wikipedia.org/wiki/OLAP_cube
- http://en.wikipedia.org/wiki/Online_analytical_processing
- http://mondrian.pentaho.com/documentation/aggregate_tables.php

Pentaho:

- <http://etl-tools.info/en/pentaho/kettle-Spoon.htm>
- <http://etl-tools.info/en/pentaho/kettle-pan.htm>
- http://kb.pentaho.com/kb/kbroot/Videos/pdi_schemaworkbench_tutorial/index.html
- <http://rpbouman.blogspot.com/2006/06/pentaho-data-integration-kettle-turns.html>
- [http://wiki.pentaho.com/display/EAI/Latest+Pentaho+Data+Integration+\(aka+Kettle\)+Documentation](http://wiki.pentaho.com/display/EAI/Latest+Pentaho+Data+Integration+(aka+Kettle)+Documentation)
- <http://wiki.pentaho.com/display/PEOpen/BI+Server+architecture+and+philosophy>
- <http://wiki.pentaho.com/pages/viewpage.action?pageId=13175232>
- http://www.schaffter.com/index.php/HowTo_setup_a_Pentaho_repository
- http://www.pentaho.com/products/demos/data_integration_fundamental_tutorial
- <http://www.pentaho.com/products/demos/showNtell.php?tab=demos&article=pentaho-architecture-overview>

Anexo D – Manual de utilização SiagETL



G E D I

MANUAL SIAGETL

REFERÊNCIA: MANUAL DE UTILIZAÇÃO SIAGETL

PARA: NUNO RODRIGUES

AUTOR:	JOÃO TIAGO RIBEIRO MENDES NATÁLIO	DATA:	10-06-11	REVISOR:	NUNO RODRIGUES	DATA:	DD/MM/AAAA	VERSÃO:	(1.0)
VERSÃO INICIAL									

DOCUMENTO PARA USO INTERNO E EXCLUSIVO DA GEDI

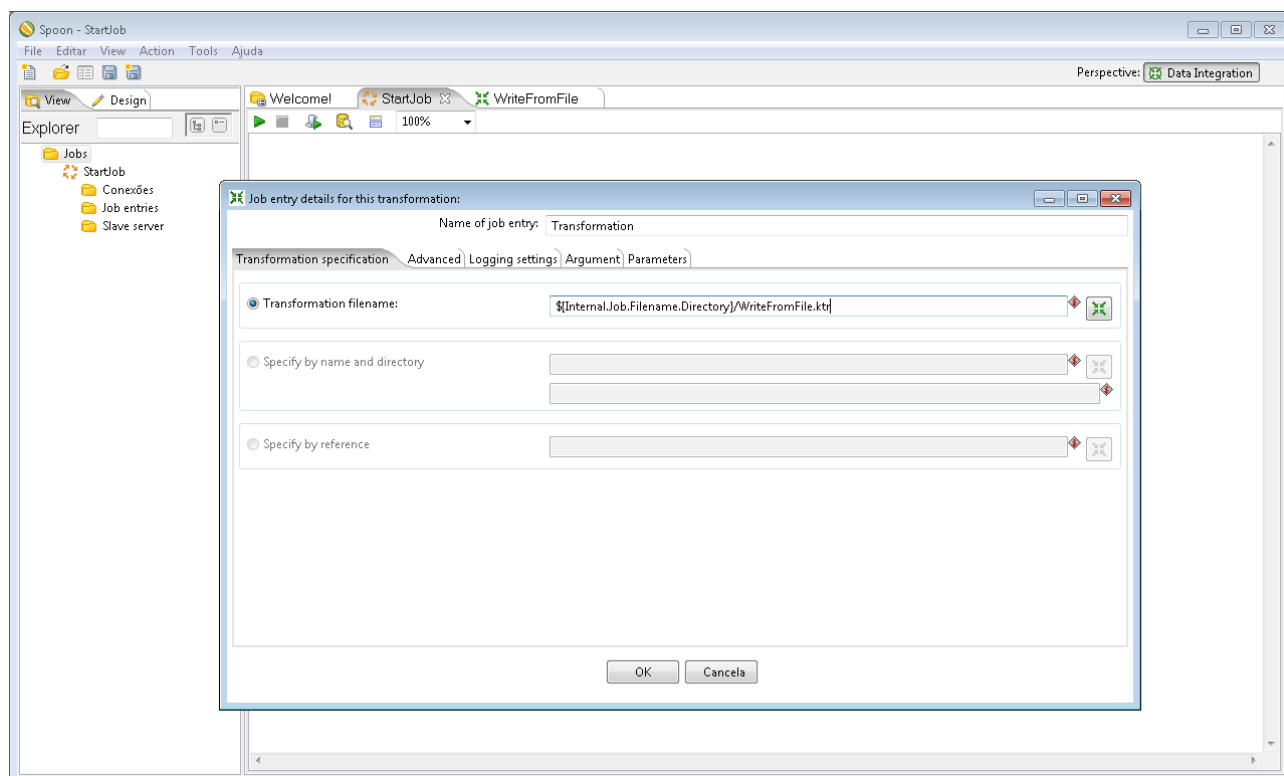
ÍNDICE

1	- INTERFACE SPOON E REGRAS GERAIS DE UTILIZAÇÃO	2
2	- INTERFACE SIAG – GESTÃO DE TRANSFORMAÇÕES ETL	4
3	- INTERFACE SIAGETL – DASHBOARD DO SERVIDOR	7
4	- INTERFACE DE CALENDARIZAÇÃO DE TRANSFORMAÇÕES ETL	9
5	- JBPM – AGENDAR A EXECUÇÃO DE UMA TRANSFORMAÇÃO	14



1 - INTERFACE SPOON E REGRAS GERAIS DE UTILIZAÇÃO

A criação de uma transformação ETL é feita através da ferramenta *Spoon* do pacote de software *Pentaho Data Integration*. Para iniciar o uso da sua interface e aprender como começar a criar transformações de ETL deve consultar o link <http://wiki.pentaho.com/display/EAI/Spoon+User+Guide>.



Como cuidados gerais antes da execução de uma transformação no servidor SiagETL, há que ter em atenção a hierarquia de ficheiros de transformação gerados pela ferramenta. Para que a transformação possa executar com sucesso através do SiagETL, todos os ficheiros criados devem estar na mesma pasta local do cliente, sem qualquer tipo de sub-hierarquia de pastas. A transformação deve ser primeiro testada localmente para garantir o seu sucesso quando enviada para o SiagETL. Após ter sido testada, há que ter diversos cuidados:

- As referências para ficheiros de outras transformações (*transformations*) e tarefas (*jobs*) dentro das tarefas criadas devem ser referidas através da variável `$[Internal.Job.Filename.Directory]/[NOME_DO_FICHEIRO]` como mostra a figura.
- No caso de haver passos (*steps*) que tenham sido partilhados entre os vários ficheiros de uma transformação ETL é necessário que envie também o ficheiro *shared.xml* presente na pasta de instalação do *Pentaho Data Integration* (dentro da pasta *.kettle*), juntamente com os ficheiros da transformação na interface do SiagETL. Este ficheiro não deve nunca ser renomeado ao ser enviado para o servidor.
- No caso de haver passos (*steps*) que necessitem de referir outro tipo de ficheiros externos (ficheiros microsoft excel, microsoft access, xml, texto, etc.) os caminhos locais usados para testar a transformação devem ser alterados para coincidir com o caminho local onde o servidor SiagETL se encontra em execução e esses



G E D I

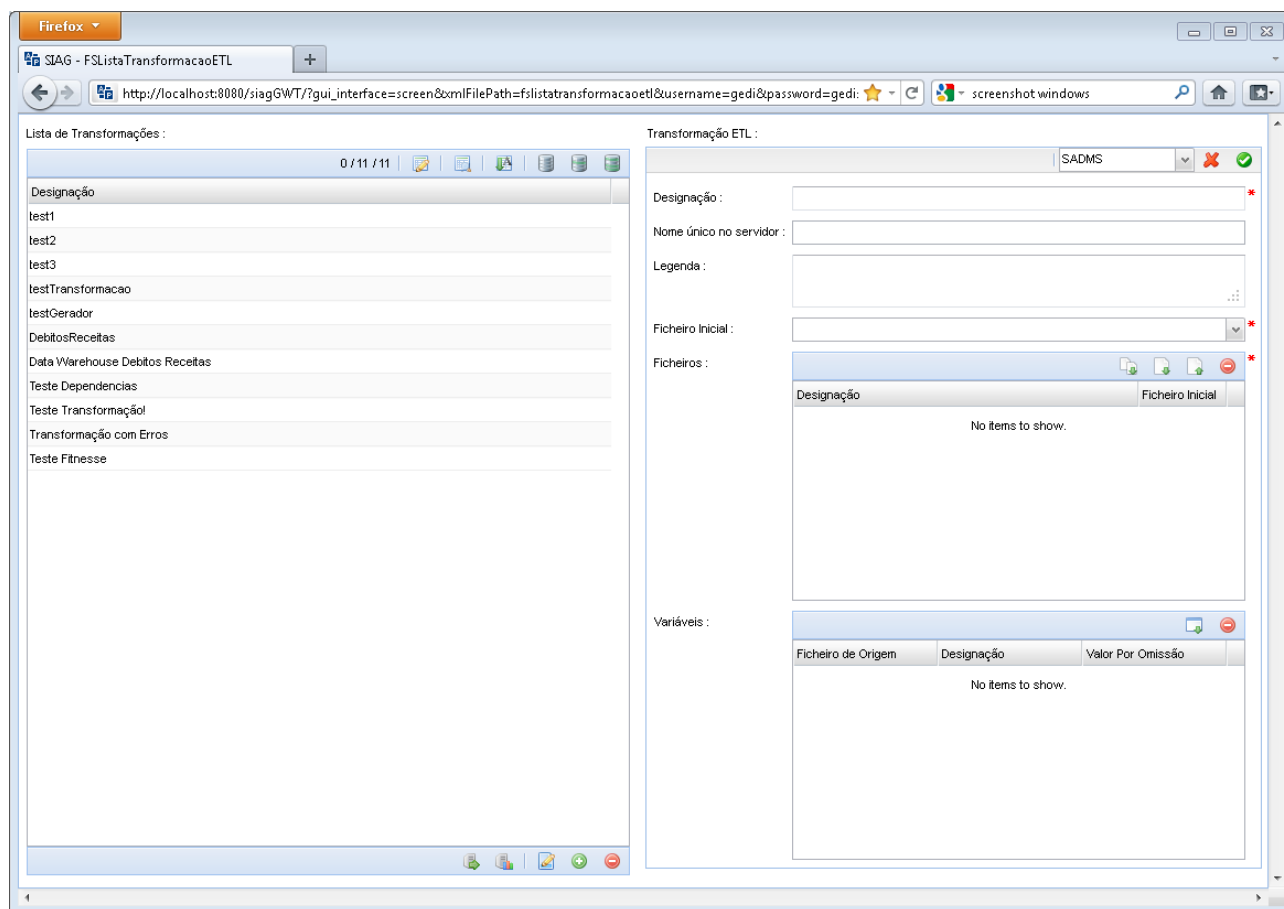
ficheiros serão guardados. Como boa prática, aconselhamos a criação de uma pasta partilhada na máquina onde se encontra o servidor SiagETL. Cabe ao administrador de sistema divulgar aos utilizadores do sistema o directório local onde esta pasta se encontra o servidor. Desta forma, antes de enviarem os ficheiros para o sistema SiagETL, devem substituir o directório local que usaram para testes para o directorio referido pelo administrador de forma a garantir que o ficheiro é encontrado quando estiver no servidor.



2 - INTERFACE SIAG – GESTÃO DE TRANSFORMAÇÕES ETL

A interface de utilização do SiagETL é composta pelos seguintes componentes:

- Uma lista com todas as transformações do sistema (lado esquerdo do ecrã).
- O ecrã de criação de transformações (lado direito do ecrã).



O componente de listagem de transformações é composto por duas barras:

- A barra superior que permite as funções comuns do SIAG de criação de filtros, ordenações e escolha de campos visíveis na lista de transformações.
- A barra inferior, composta por diversas funções abaixo descritas.



Componentes da barra inferior da lista de transformações:





Ícone	Descrição Breve	Descrição Detalhada
	Envio para o servidor	Envia o conjunto de transformações ETL seleccionadas na lista de transformações para o servidor remoto SiagETL.
	Abrir dashboard do servidor	Abre uma janela com o dashboard do servidor de ETL, permitindo a gestão e execução de transformações ETL no servidor.
	Edição de transformação ETL	Permite a edição da transformação ETL seleccionada na lista de transformações. Como atalho pode também fazer duplo-clique sobre uma transformação para abrir o modo de edição.
	Criar nova transformação ETL	Limpa o ecrã do lado esquerdo para permitir a introdução de uma nova transformação ETL no SIAG.
	Remover transformação ETL	Remove as transformações ETL que estiverem seleccionadas na lista de transformações.

No ecrã esquerdo de criação / edição de transformações ETL podemos verificar os seguintes campos:



Nome do Campo	Descrição
Designação	Nome dado a uma determinada transformação ETL no sistema SIAG.
Nome único no servidor	Nome automaticamente gerado pelo SIAG pelo qual será conhecida a transformação ETL quando for enviada para o servidor. É este nome que será visto quando for gerir as transformações através do dashboard do servidor.
Legenda	Um campo de texto livre para auxiliar o utilizador a inserir comentários sobre uma transformação ETL.
Ficheiro inicial	O ficheiro que dá início à execução de uma transformação ETL. Deve seleccionar o ficheiro da tarefa (<i>job</i>) que contém o passo (<i>step</i>) START entre os vários ficheiros que compõe a transformação ETL. Caso a transformação ETL seja apenas composta por um único ficheiro de transformação (<i>transformation</i>) deve seleccioná-lo.
Ficheiros	O componente de envio e recepção dos ficheiros da transformação ETL modelada através da ferramenta <i>Spoon</i> . Para detalhes sobre os componentes desta lista consulte "Listagem de componentes do campo <i>Ficheiros</i> " deste manual.
Variáveis	Permite a extracção e substituição dinâmica de variáveis designadas nos diversos ficheiros que compõe a transformação ETL. Para detalhes sobre os componentes desta lista consulte as listagens seguintes deste manual.



Descrição dos componentes do campo *Ficheiros*:

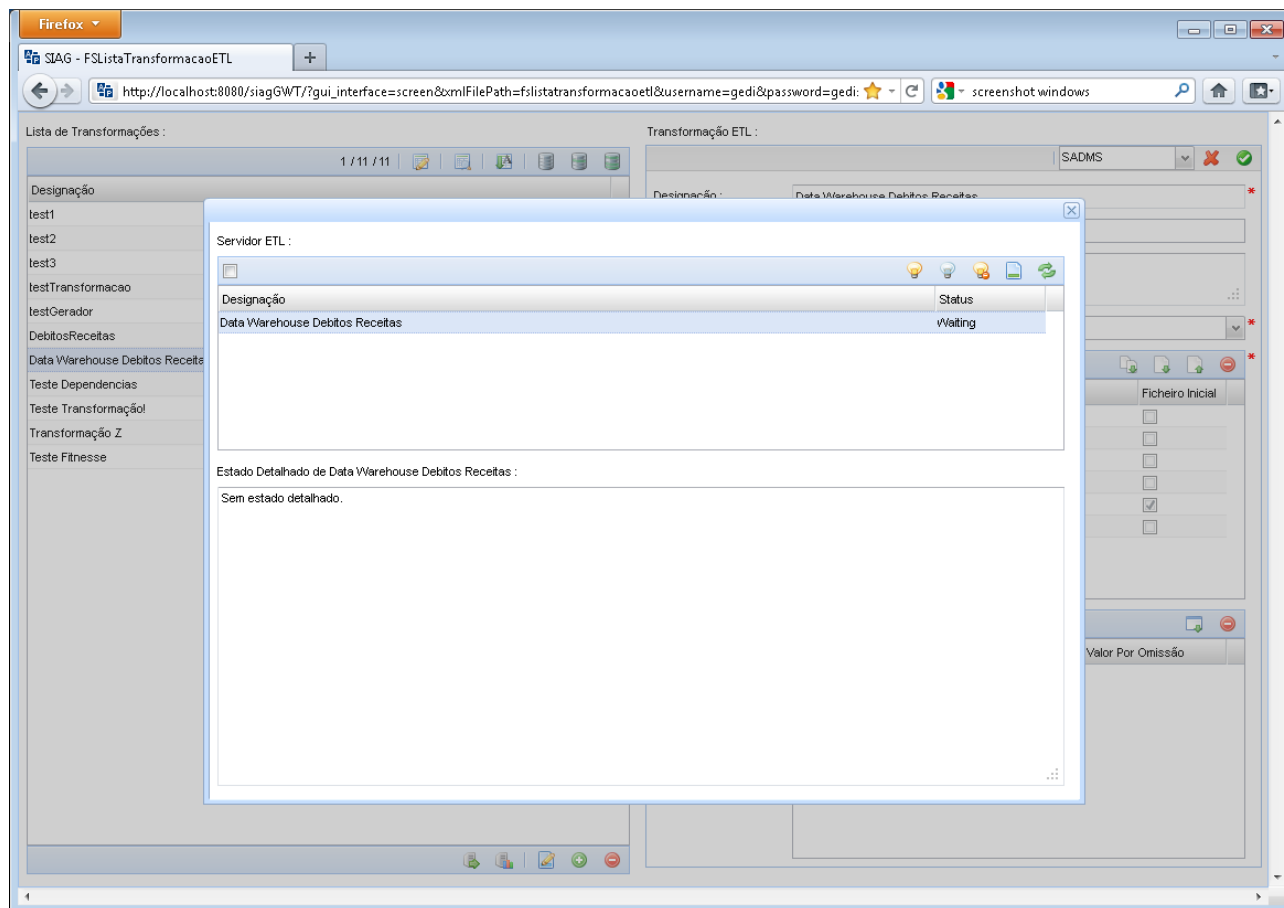
Ícone	Descrição Breve	Descrição Detalhada
	Importação de múltiplos ficheiros.	Permite o envio de vários ficheiros para o SIAG. Após ter testado a transformação ETL e a ter preparado para execução conforme descrito no capítulo 1 deste manual, pode enviar os vários ficheiros da transformação criada através deste botão.
	Alterar linha.	Permite alterar o ficheiro presente numa linha seleccionada da lista de ficheiros da transformação ETL, alterando o seu conteúdo.
	Exportar ficheiro.	Permite o download de um ficheiro seleccionado da lista de ficheiros.
	Remover ficheiro.	Remove o ficheiro que estiver seleccionado na lista de ficheiros.

Descrição dos componentes do campo *Variáveis*:

Ícone	Descrição Breve	Descrição Detalhada
	Extrair variáveis.	Permite a extracção de variáveis dos ficheiros que compõe a transformação ETL. Caso tenha definido um passo (<i>step</i>) do tipo “ <i>Set Variables</i> ” em algum dos ficheiros que compõe a transformação, este componente irá ler os valores descritos nesse passo e permitir a sua substituição dinâmica sempre que uma transformação ETL é enviada para o servidor para execução. Note que esta substituição é feita dinamicamente, nunca afectando os ficheiros em si.
	Remover variável.	Remove uma variável seleccionada da lista de variáveis. Note que a remoção de uma variável desta lista não afecta a declaração de variáveis nos ficheiros da transformação em si, apenas impedem a modificação dinâmica do seu valor quando enviada para o SiagETL.



3 - INTERFACE SIAGETL – DASHBOARD DO SERVIDOR









É através deste dashboard que podemos controlar a execução, paragem e remoção de transformações no servidor SiagETL. Esta interface é composta por dois componentes principais:

Nome do Campo	Descrição
Servidor ETL	<p>É nesta lista que são mostradas as transformações ETL enviadas a partir da interface do SIAG. Nesta lista podemos encontrar o campo <i>Descrição</i> com o nome único gerado pelo SIAG e o campo <i>Status</i> com a descrição breve do estado da transformação ETL. Também é possível obter o estado detalhado de uma transformação ETL fazendo duplo-clique sobre ela.</p> <p>Os estados possíveis são:</p> <ul style="list-style-type: none">• <i>Waiting</i> – Aguardando por uma ordem do servidor.• <i>Halting</i> – A transformação já executou e encontra-se em paragem.• <i>Stopped</i> – A transformação foi interrompida pela ordem de um utilizador.• <i>Stopped (with errors)</i> – A transformação parou a sua execução devido a erros.• <i>Running</i> – A transformação encontra-se em execução.• <i>Finished</i> – A transformação terminou com sucesso a sua execução.
Estado Detalhado	<p>É neste campo que é mostrado em detalhe as últimas linhas de actividade de uma determinada transformação ETL presente no servidor.</p>



Descrição dos componentes do campo *Servidor ETL*:

Ícone	Descrição Breve	Descrição Detalhada
	Actualização automática do estado do servidor.	Este componente quando seleccionado permite a actualização automática da lista de transformações com o estado do servidor de 10 em 10 segundos. Note que a selecção desta opção implica a desactivação do botão de refrescar estado do servidor.
	Execução de transformação.	Permite inicializar a execução das transformações ETL seleccionadas na lista de transformações.
	Paragem de execução de transformação.	Permite interromper a execução das transformações ETL seleccionadas na lista de transformações.
	Remoção de transformação.	Permite a remoção das transformações ETL seleccionadas na lista de transformações do servidor.
	Obter estado detalhado.	Obtém as últimas linhas de actividade de uma determinada transformação ETL seleccionada na lista de transformações do servidor.
	Refrescar estado do servidor.	Pede para actualizar o estado das transformações ETL na lista de transformações.



4 - INTERFACE DE CALENDARIZAÇÃO DE TRANSFORMAÇÕES ETL

Firefox

SIAG - FSSchedulerETL

http://localhost:8080/siagGWT/?gui_interface=screen&xmlFilePath=fsschedulereti&use

Transformação ETL : *

Data Warehouse Debitos Receitas

Tempo limite de Execução (horas) : *

8

Hora de Execução : * Data de Início : *

10:00:00 2011/12/31

Tipo : * Data de Fim : *

Agendar uma única vez

O agendamento de execução de uma transformação ETL é feito ao criar uma instância do processo *SchedulerETL* na área *O meu trabalho* na interface do sistema SIAG-AP. Neste processo podemos agendar a execução de uma transformação consoante uma determinada calendarização definida pelo utilizador.

O sistema SiagETL, no momento definido através desta interface, irá:

- Enviar a transformação ETL para o servidor SiagETL.
- Inicializar a sua execução no servidor.
- Esperar até que a execução tenha sido terminada com ou sem sucesso ou até um determinado período máximo de tempo também definido pelo utilizador através da interface.
- Remover a transformação do servidor.



Tendo em conta estes factores, existem algumas condições a considerar:

- Se no momento de execução da transformação ETL escolhida a transformação já se encontrar no servidor SiagETL, o utilizador irá ser notificado que ocorreu um erro no envio da transformação ETL para o servidor e esta não irá ser executada. Para ser executada com sucesso a transformação agendada não pode estar presente no servidor na hora de execução.
- Se o tempo máximo de execução definido pelo utilizador for atingido, a execução da transformação ETL irá produzir um erro em que diz que o tempo máximo de execução foi atingido. Este parâmetro serve como precaução para a execução de transformações ETL mal definidas que bloqueiem o servidor e deve ser definido tendo em conta o tempo médio de execução da transformação ETL quando foi testada.
- A calendarização válida para o processo é a que o utilizador está de momento a ver no ecrã quando for a submeter o formulário no SIAG.

Nesta interface de agendamento podemos encontrar os seguintes campos:

Nome do Campo	Descrição
Transformação ETL	É neste campo que seleccionamos a transformação ETL já guardada no SIAG-AP que queremos agendar para execução futura.
Tempo limite de execução (horas)	É neste campo que é definido o limite máximo em horas que o servidor SiagETL irá esperar que a transformação ETL termine a sua execução como forma de prevenir o bloqueio do servidor na execução de uma transformação mal definida.
Hora de Execução	A hora em que a transformação irá executar.
Data de Execução	A data em que a transformação irá executar.
Data de Fim (opcional)	A data em que a calendarização definida perde efeito.
Tipo	<p>O tipo de calendarização que se quer definir. Iremos ver em mais detalhes cada uma destas opções seguidamente.</p> <p>Neste campo podemos escolher:</p> <ul style="list-style-type: none">• Agendar uma única vez• Agendar em minutos• Agendar em horas• Agendar em dias• Agendar em semanas• Agendar em meses



TIPOS DE CALENDARIZAÇÃO

AGENDAR UMA ÚNICA VEZ

Agenda uma única vez a execução da transformação ETL definida na hora e data de início na interface.

AGENDAR EM MINUTOS / HORAS / DIAS

Exemplo:

Tipo : * Data de Fim :

Agendar em dias *

Intervalo em Dias : *

10

Agenda uma transformação dado um intervalo de um determinado tipo, com a hora e data de início definido na interface.

AGENDAR EM SEMANAS

Exemplo:

Tipo : * Data de Fim :

Agendar em semanas *

Intervalo em Semanas : *

1

Dias da Semana
Segunda-Feira
Terça-Feira
Quarta-Feira
Quinta-Feira
Sexta-Feira
Sábado
Domingo

Dias Selecionados
Segunda-Feira
Quarta-Feira
Sábado

Arrastando os dias de semana do lado direito do ecrã para a lista do lado esquerdo do ecrã definimos os dias da semana em que queremos que uma determinada transformação seja executada. Para além da definição dos dias da semana, ainda podemos definir um intervalo em semanas em que vai ser executado.



AGENDAR EM MESES

Nesta opção, começamos por escolher quais os meses em que queremos que uma determinada transformação ETL execute, arrastando os meses presentes na lista do lado direito do ecrã para a lista do lado esquerdo do ecrã.

Após definidos os meses, dentro deste tipo de calendarização temos de definir quais os dias do mês em que queremos executar a transformação, optando por uma das opções do campo *Definição de dias*:

- Definir por dias do mês
- Definir por dias da semana

Na opção *Definição por dias do mês*, escolhemos os dias do mês em que queremos que execute a transformação, conforme o exemplo:

The screenshot displays the 'SIAG - FSSchedulerETL' application window. The 'Tipo' dropdown is set to 'Agendar em meses'. The 'Data de Fim' field is empty. The 'Meses' list on the left contains all months from Janeiro to Dezembro. The 'Meses Seleccionados' list on the right contains Janeiro, Abril, Junho, Agosto, Outubro, and Dezembro. The 'Definição de dias' dropdown is set to 'Definir por dias do mês'. The 'Dias do Mês' list on the left contains the numbers 22 through 30, along with 'Último Dia do Mês'. The 'Dias Seleccionados' list on the right contains 3, 7, 11, and 'Último Dia do Mês'.

De notar que o utilizador é que é responsável por escolher os dias correctos de um determinado mês. Se optar por exemplo por executar uma transformação ETL no dia 30 do mês de Fevereiro, esse dia será ignorado.



Na opção *Definição por dias da semana*, escolhemos os dias do mês em que queremos que execute a transformação em relação às semanas do mês, conforme o exemplo:

The screenshot shows the SIAG - FSSchedulerETL application window. The interface is divided into several sections:

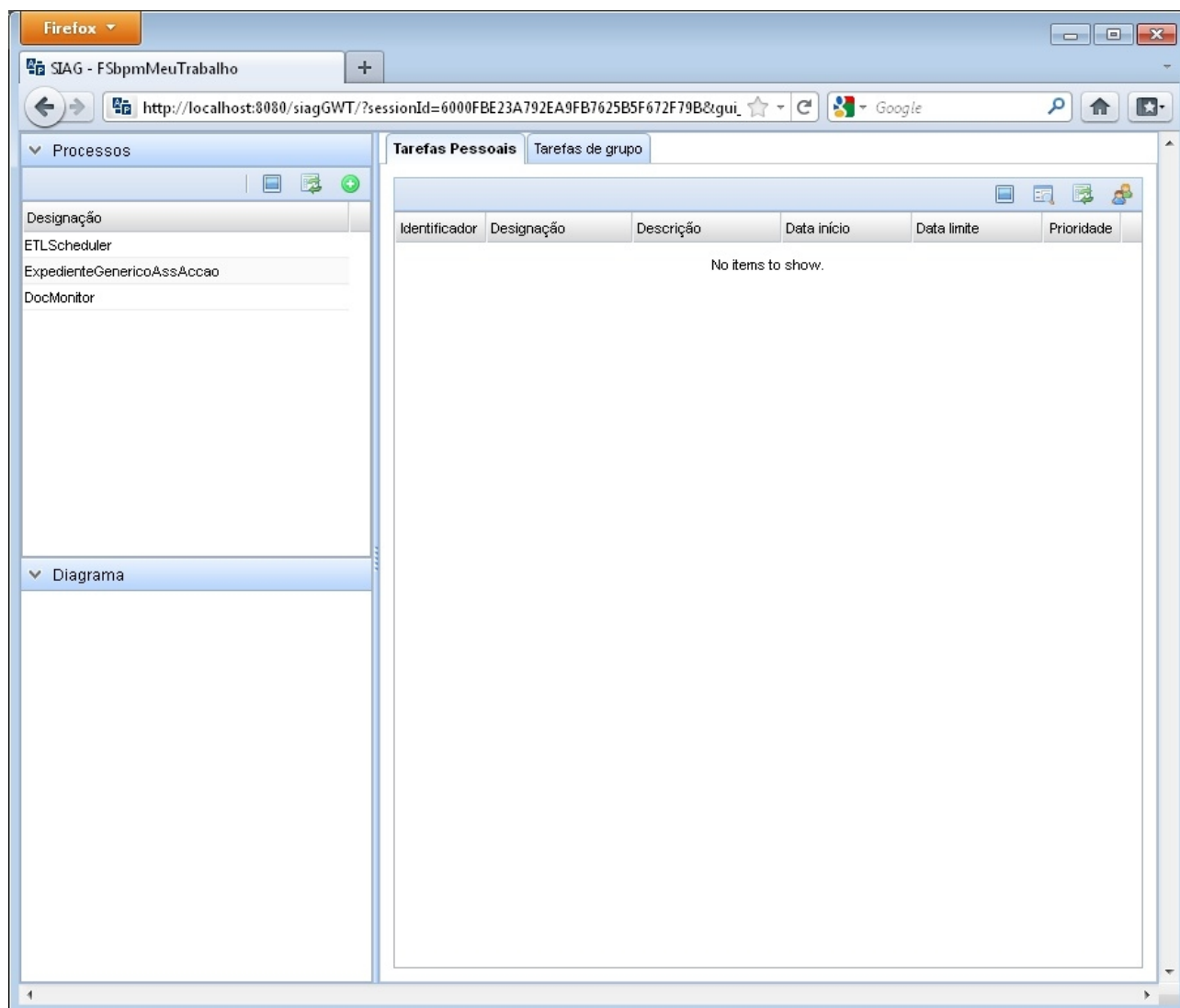
- Agendar em meses:** A dropdown menu with a calendar icon.
- Meses:** A list of months from Janeiro to Dezembro. Janeiro is selected.
- Meses Seleccionados:** A list showing the selected months: Janeiro, Abril, Junho, Agosto, Outubro, and Dezembro.
- Definição de dias:** A dropdown menu with the option "Definir por dias da semana" selected.
- Prefixo:** A dropdown menu with the option "Última(o)" selected.
- Dias da Semana:** A list of days from Segunda-Feira to Domingo. Domingo is selected.
- Dias Seleccionados:** A list showing the selected days: Primeira(o) Segunda-Feira, Segunda(o) Segunda-Feira, and Última(o) Domingo.



5 - JBPM – AGENDAR A EXECUÇÃO DE UMA TRANSFORMAÇÃO

Agendar a execução de uma transformação ETL passa por inicializar uma instância do processo *ETLScheduler*. Para tal este processo deverá ser instalado no sistema SIAG através do ecrã *Processos* que pode ser encontrado no menu *Desenvolvimento e Orquestração de Sistema* na barra lateral esquerda do ecrã principal do SIAG-AP.

Para instanciar o processo deve primeiro aceder ao ecrã *O meu trabalho* demonstrado na figura seguinte, podendo ser acedido pelo menu *Utilitários Pessoais* presente também na barra lateral do ecrã principal do SIAG-AP.





Neste ecrã, selecione o processo *ETLScheduler* e carregue no botão de *Criar Nova Instância*. Uma nova tarefa pessoal com a designação *Agendar* deverá aparecer na lista do lado direito do écran. Ao abrir esta tarefa, irá ser aberto um novo ecrã de agendamento da transformação ETL como demonstra o exemplo da figura seguinte:

Firefox

SIAG - FSbpmTaskWrapper

http://localhost:8080/siagGWT/?sessionId=6000FBE23A792EA9FB7625B5F672F79B&gui

Caminho: Iniciar

Transformação ETL :

Data Warehouse Debitos Receitas

Tempo limite de Execução (horas) :

1

Hora de Execução :

10:00:00

Data de Início :

2011/06/14

Tipo :

Agendar uma única vez

Data de Fim :

Atributos :

name	value
Prioridade	0

Adicionar Utilizadores/Grupos :



Após a definição das opções de agendamento para a transformação ETL escolhida, deverá seleccionar no topo do ecrã o caminho *iniciar* para iniciar o processo de execução de transformações e submeter o formulário.

Note que no ecrã *O meu trabalho* a sua tarefa pessoal mudou para o estado *Monitor* demonstrado na figura seguinte, onde poderá redefinir o agendamento da transformação enquanto a data limite definida anteriormente não for atingida.

Identificador	Designação	Descrição	Data início	Data limite	Prioridade
650013	Monitor		2011/06/14		0

Caso a execução da transformação ETL falhe por algum motivo, o utilizador será notificado através de uma nova tarefa pessoal *Erro* que irá surgir neste ecrã.